UNIVERSIDADE DE SANTA CRUZ DO SUL
CURSO DE CIÊNCIA DA COMPUTAÇÃO

Ederson Luis Posselt

APLICAÇÃO EMBARCADA NÃO INTRUSIVA DE CLASSIFICAÇÃO DE CONDUTORES

Santa Cruz do Sul, dezembro de 2008.

UNIVERSIDADE DE SANTA CRUZ DO SUL
CURSO DE CIÊNCIA DA COMPUTAÇÃO

Ederson Luis Posselt

APLICAÇÃO EMBARCADA NÃO INTRUSIVA DE CLASSIFICAÇÃO DE CONDUTORES

Trabalho de conclusão apresentado ao Curso de Ciências da Computação da Universidade de Santa Cruz do Sul, para a obtenção do título de Bacharel em Ciências da Computação.

Orientador: Prof. Dr. Rolf Fredi Molz

Co-orientadora: Profª. Dra. Rejane Frozza

Santa Cruz do Sul, dezembro de 2008.

"*Imagination is more important than knowledge.*
Knowledge is limited.
Imagination encircles the world."
Albert Einstein.

"*A mente que se abre a uma nova idéia*
jamais voltará ao seu tamanho original."
Albert Einstein.

AGRADECIMENTO

Agradeço imensamente a minha mãe Glaci, minha avó Similda e a minha noiva Kéli. As três mulheres da minha vida que me apoiaram nesses longos anos. Seus estímulos e carinho foram as armas desta vitória.

Agradeço a todos os meus familiares, em especial a Flávia, Genésio, Eduarda, Jonas, Sérgio, Zuleica, que entenderam minhas ausências. Por acreditarem na concretização deste sonho, compartilho, agora, o mérito desta conquista.

A todos os meus colegas de turma, em especial aos amigos Arthur, Geovane, Josué e Ricardo. Mais do que colegas de classe, mais do que parceiros no estudo das temíveis aulas de Formais e Compiladores. Irmãos que fizeram parte de um cotidiano, de cada dia de nossa caminhada que acaba aqui, como alunos por força natural, mas que não acabará como amigos por força de sentimento.

A meus colegas de trabalho que me acompanharam e me incentivaram nessa jornada, especialmente o Fabiano Horn, que partilhou de seu conhecimento, abnegando muitas vezes de seu tempo para me ajudar. A toda equipe da empresa em que trabalho, Imply Tecnologia Eletrônica, que muitas vezes entenderam as minhas necessidades e me apoiaram com tempo e recursos.

Enfim, muito obrigado ao corpo docente do curso de computação, que compartilharam de seus conhecimentos, especialmente ao meu orientador, professor, colega e amigo Rolf Fredi Molz, que me acompanhou no decorrer desse projeto, contribuindo para o sucesso do mesmo.

E a todos aqueles que contribuíram na busca deste ideal, meus mais profundos e sinceros agradecimentos.

RESUMO

O trânsito representa hoje um grave problema tanto nas esferas federal, estadual como municipal, acometendo em especial, segundo índices estatísticos do Departamento Nacional de Trânsito (DENATRAN, 2007), jovens de todo o mundo, devido a sua maior vulnerabilidade e exposição aos riscos de mortes e de acidentes no trânsito. No Brasil mais de 40.000 pessoas perdem a vida anualmente em acidentes nas estradas, porém, acredita-se que estes números sejam ainda mais relevantes, haja vista que as estatísticas de trânsito continuam falhas.

De acordo com o Departamento Estadual de Trânsito do Rio Grande do Sul (DETRAN-RS, 2007), o erro humano, em todo o mundo, é responsável por mais de 90 % dos acidentes registrados, tendo como principais imprudências determinantes de acidentes fatais no Brasil, por ordem de incidência: a velocidade excessiva; dirigir sob efeito de álcool; distância insuficiente em relação ao veículo dianteiro; desrespeito à sinalização; dirigir sob efeito de drogas e assim por diante.

Baseado nos altos índices de acidentes de trânsito, apresenta-se um projeto de um sistema embarcado não intrusivo na tentativa de reduzir esses índices, tendo como informação principal, o fato de 90% ou mais dos acidentes serem ocasionados por erro humano. Para o desenvolvimento do projeto foram utilizados os seguintes itens tecnológicos: processador ARM7, receptor GPS e lógica *fuzzy* no processo de classificação do condutor. Com a associação desses itens foi possível desenvolver o presente sistema que classifica o condutor e se posiciona de forma pró-ativa diante das atitudes do motorista no volante.

Palavras-chave: Trânsito, sistema inteligente, processador ARM7, receptor GPS, lógica fuzzy.

ABSTRACT

Traffic is now a serious problem in the federal spheres, state and municipal levels, affecting in particular young people around the world due to their higher vulnerability and exposure to the risks of deaths and injuries in traffic (DENATRAN, 2007). In Brazil more than 40,000 people lose their lives annually in road accidents, but it is believed that the figures are even more relevant, is seen that the statistics of transit failures continue.

Worldwide, human error is responsible for more than 90% of accidents recorded. This estimate is provided by State Department of Transit of Rio Grande do Sul (RS-DETRAN, 2007). The main dangers points are: the excessive speed, driving under influence of alcohol; distance inadequate in relation to the vehicle front, ignoring the signs, driving under influence of drugs and so on.

Based on the high rates of traffic accidents this document shows a system not intrusive to try to reduce these indices, with the main information, the fact that 90% or more of the accidents are caused by human error. For the development of this project was used the following technology: ARM7 processor, GPS receiver and fuzzy logic in the process of classifying the driver. With the combination of these items was possible to develop a system that classifies the driver and has a proactive driver's attitude.

Keywords: Transit, smart systems, ARM7 processor, GPS receiver, fuzzy logic.

LISTA DE FIGURAS

Figura 1 - Diagrama em blocos do núcleo ARM7TDMI-S 18
Figura 2 – Estrutura de *pipeline* do processador ARM 19
Figura 3 - Exemplo do uso de registradores 19
Figura 4 – Os sete modos de operação do processador ARM7 20
Figura 5 - Duas formas de tratamento das variáveis. *Little-endian*/ *Big-endian* 21
Figura 6 – Distribuição dos pinos do processador ARM7 LPC2138 23
Figura 7 - Sistema de barramentos ARM7 24
Figura 8 - Mapa da memória processador ARM. 25
Figura 9 - Kit de desenvolvimento MCB2130 29
Figura 10 - Diagrama de blocos do kit de desenvolvimento MCB2130 30
Figura 11 - IDE de desenvolvimento *MicroVision* da Keil Software 32
Figura 12 - Principais configurações de um projeto utilizando a IDE *MicroVision* 33
Figura 13 – Aplicativo *Flash Utility* LPC2000 desenvolvido pela *Philips Semiconductors* ... 35
Figura 14 - Aplicativo Flash Magic 36
Figura 15 - Ilustração dos satélites em órbita 39
Figura 16 - Cálculo da posição do receptor GPS 41
Figura 17 - GPS Receiver Engine Board ET-102/ER-102 43
Figura 18 - Cálculo do checksum na linguagem C 45
Figura 19 - Diferença entre lógica clássica e lógica *fuzzy* 54
Figura 20 - Exemplo do uso de conjuntos *fuzzy* 55
Figura 21 - Exemplo de fuzzificação 56
Figura 22 - Diagrama de blocos de um controlador *fuzzy* típico 57
Figura 23 - Exemplo de regra de produção *fuzzy* 58
Figura 24 - Estrutura do projeto de classificação de condutores 63
Figura 25 - Configuração do *clock* de processamento 65
Figura 26 - Tabela de configuração do registrador PLLCFG. 66
Figura 27 - Código fonte de inicialização serial 69
Figura 28 - Código fonte responsável pelo envio serial 70
Figura 29 - Rotina de recebimento serial 71
Figura 30 - Exemplo de transmissão TTL e RS232 72
Figura 31 - Conversor serial TTL para serial RS232 73

Figura 32 - Estrutura utilizada para capturar os dados do receptor GPS.....74
Figura 33 - Sistema de análise dos dados para geração das métricas do sistema.....75
Figura 34 - Gráfico de Velocidade por tempo.....76
Figura 35 - Gráfico da distribuição das velocidades 77
Figura 36 - Gráfico de aceleração e desaceleração.....78
Figura 37 - Gráfico de aceleração agressiva.....79
Figura 38 - Gráfico da constância de velocidade 80
Figura 39 - Gráfico comparativo entre a constância em percurso rodoviário e urbano 81
Figura 40 - Entradas do sistema fuzzy.....86
Figura 41 - Diagrama de Hassi-Euler dos conjuntos AP, AM, AA e AMA. 88
Figura 42 - Diagrama de Hassi-Euler dos conjuntos VD, VM, VR e VMR. 90
Figura 43 - Diagrama de Hassi-Euler dos conjuntos DP, DM, DA e DMA. 91
Figura 44 - Diagrama de Hassi-Euler dos conjuntos CMRU, CMRE, CMB e CMMB.....92
Figura 45 - Gráfico de pertinência da saída fuzzy.....95
Figura 46 - Visualização da saída *fuzzy* através dos LED`s 97
Figura 47 - Parte do código da biblioteca LPC213x.h 103
Figura 48 - Código fonte do primeiro ensaio com a placa MCB2130 104
Figura 49 - Código fonte utilizando interrupção FIQ.....106

LISTA DE TABELAS

Tabela 1 - Lista das principais interrupções 26
Tabela 2 - Comparativo entre alguns processadores ARM fabricados pela Philips 27
Tabela 3 - Configuração da placa MCB2130 através de *jumpers* 29
Tabela 4 - Principais características do módulo receptor GPS ET-102/ER-102 42
Tabela 5 – Distribuição de pinos para acesso ao módulo receptor GPS ET-102/ ER-102 44
Tabela 6 - Formato GGA do protocolo NMEA 46
Tabela 7 - Formato GLL do protocolo NMEA 47
Tabela 8 - Formato GSA do protocolo NMEA 48
Tabela 9 - Formato GSV do protocolo NMEA 49
Tabela 10 - Formato RMC do protocolo NMEA 50
Tabela 11 - Formato VTG do protocolo NMEA 51
Tabela 12 - Comparativo entre TTL e RS232 71
Tabela 13- Análise dos percursos urbanos quanto a média e máxima de velocidade. 82
Tabela 14 - Análise dos percursos secundários quanto a média e máxima de velocidade. 82
Tabela 15 - Análise dos percursos rodoviários quanto a média e máxima de velocidade. 83
Tabela 16 - Funções de pertinência da variável de aceleração 88
Tabela 17 - Funções de pertinência da variável de velocidade 89
Tabela 18 - Funções de pertinência da variável de desaceleração 90
Tabela 19 - Funções de pertinência da variável de classificação do motorista 91
Tabela 20 - Regras fuzzy de velocidade 93
Tabela 21 - Regras *fuzzy* de aceleração e desaceleração 94
Tabela 22 - Esquema da pontuação do condutor 97
Tabela 23 - Identificação do pressionamento do botão INT1 105
Tabela 24 - Configuração do acionamento dos LED`s 105

LISTA DE ABREVIATURAS

DENATRAN	Departamento Nacional de Trânsito
DETRAN-RS	Departamento Estadual de Trânsito do Rio Grande do Sul
UNISC	Universidade de Santa Cruz do Sul
CISC	*Complex Instruction Set Computer*
RISC	*Reduced Instruction Set Computer*
IDE	*Integrated Development Environment*
I/O	*Input/ Output*
ARM	*Advanced RISC Machine*
AHB	*Advanced High Performance Bus*
APB	*ARM Peripheral Bus*
IRQ	*Vectored Interrupt Request*
FIQ	*Fast Interrupt Request*
VIC	*Vectored interrupt Controller*
GPS	*Global Positioning System*
SPS	*Standard Positioning Service*
PPS	*Precise Positioning Service*
OEM	*Original equipment manufacturer*
HEXA	Hexadecimal
LED	*Light Emitter Diode*
JTAG	*Joint Test Action Group*
UART	*Universal asynchronous receiver/transmitter*
NMEA	*National Marine Electronics Association*

SUMÁRIO

RESUMO 5
ABSTRACT 6
INTRODUÇÃO 13
1 PROCESSADOR ARM 15
1.1 Processadores CISC e RISC 15
1.2 História do processador ARM 16
1.3 Núcleo do processador ARM7 17
1.4 Características do Processador ARM7 LPC213x 21
1.5 Pinagem do processador ARM7 LPC2134/ 2136/ 2138 22
1.6 Barramentos do processador ARM7 LPC2138 23
1.7 Mapa de memória do processador ARM7 24
1.8 Sistema de interrupções do processador ARM7 LPC2138 25
1.9 Comparativo técnico entre alguns processadores ARM 27
2 SOFTWARE DE DESENVOLVIMENTO 28
2.1 Kit de desenvolvimento MCB2130 28
2.2 Hardware MCB2130 30
2.3 Ferramenta de desenvolvimento *MicroVision* 3 31
3 APLICATIVOS DE GRAVAÇÃO DA MEMÓRIA FLASH 34
3.1 Aplicativo *Philips Flash Utility* LPC2000 34
3.2 Aplicativo *Flash Magic* 35
4 SISTEMA DE POSICIONAMENTO GLOBAL - GPS 38
4.1 História do GPS 38
4.2 Funcionamento do GPS 38
4.3 Receptores GPS 41
4.4 *GPS Receiver Engine Board* ET-102/ER-102 42
4.5 *NMEA Output Command* 44
4.5.1 GGA-*Global Positioning System Fixed Data* 45
4.5.2 GLL-*Geographic Position-Latitude/Longitude* 46
4.5.3 GSA-*GNSS DOP and Active Satellites* 47
4.5.4 GSV-*GNSS Satellites in View* 48
4.5.5 RMC-*Recommended Minimum Specific GNSS Data* 50

4.5.6 VTG-*Course Over Ground and Ground Speed*.....51
4.6 *NMEA Imput Command* 52
5 SISTEMAS INTELIGENTES.....53
5.1 Lógica *Fuzzy*.....54
5.1.1 Teoria dos conjuntos *fuzzy*.....55
5.1.2 Variável lingüística.....56
5.1.3 Sistema de inferência *fuzzy*.....56
6 ANÁLISE COMPORTAMENTAL DOS MOTORISTAS.....59
7 DESENVOLVIMENTO DO PROJETO.....63
7.1 Estrutura do projeto 63
7.2 Configuração da velocidade de processamento.....64
7.3 Comunicação serial.....66
7.4 Análise dos dados obtidos 73
7.5 Processo de classificação do condutor.....83
7.6 Definições do algoritmo *fuzzy* na classificação do condutor.....85
7.6 Resultado dos testes realizados.....96
CONCLUSÃO.....98
REFERÊNCIAS 100
ANEXO A: ENSAIOS REALIZADOS 103

INTRODUÇÃO

Os acidentes de trânsito representam atualmente um relevante problema a nível nacional, onde milhares de pessoas perdem as suas vidas anualmente, sendo que em grande parte dos casos, pessoas jovens e produtivas, assim classificadas pelos órgãos responsáveis pelo controle do trânsito brasileiro, são vítimas deste fato infortúnio. Essas vidas são acometidas na maioria das vezes por erro do próprio condutor ou por erro de um terceiro, que simplesmente fazem com que as leis de trânsito passem despercebidas.

Muitos indivíduos ao conduzirem seus veículos, criam condições irregulares e irreversíveis para que o acidente ocorra, isto normalmente sucede em função da completa ignorância em relação aos elementos causadores dos mesmos. Os fatores que proliferam em grande intensidade nas rodovias e ruas brasileiras devem-se: a reação normal associada à distância incompatível com a velocidade, o que torna impossível a parada de emergência no momento necessário; reação retardada mais o não reconhecimento de tal situação relacionada com a distância incompatível com a velocidade, predispondo a uma parada impossível e situação irreversível; reação retardada somada ao reconhecimento da situação mais a distância incompatível com a velocidade, resultando em parada impossível (GREF, 1990).

O trabalho descrito neste documento busca o estudo e desenvolvimento de itens de segurança automotivos com a utilização de um processador ARM (*Advanced RISC Machine*) e um receptor GPS (*Global Positioning System*), gerando um perfil do condutor utilizando-se da implementação da lógica *fuzzy* sobre os dados coletados e realizando um tratamento pró-ativo quando necessário. Um dos desafios do projeto ocorreu na parte da programação, pois para desenvolver um sistema embarcado foi necessário conhecer a estrutura e as peculiaridades do processador ARM7, do receptor GPS, além de conhecer o protocolo NMEA (*National Marine Electronics Association*).

Como objetivo principal desse trabalho teve-se o desenvolvimento de um sistema que vise a redução dos índices de acidentes caracterizados pela imprudência e falta de atenção no trânsito. Para desenvolver a aplicação proposta, foi necessário a formulação de um algoritmo inteligente capaz de gerar a classificação do condutor baseado em pesquisas realizadas na área psicológica. Os objetivos específicos alcançados nesse trabalho foram: estudo da arquitetura

dos processadores ARM7, funcionamento do receptor GPS, protocolo NMEA, técnicas inteligentes na classificação e identificação do estado do condutor e entendimento da psicologia do trânsito.

A organização do trabalho está dividida em capítulos que contemplam a evolução do projeto, sendo assim, os capítulos iniciais tratam da fundamentação teórica e na sua seqüência são apresentados os capítulos sobre o desenvolvimento e implementação do projeto. Os capítulos relacionados ao longo desse trabalho contemplam os seguintes conteúdos: processador ARM, kit de desenvolvimento Keil, software de atualização do firmware do processador ARM, ensaios realizados sobre os equipamentos envolvidos no projeto, GPS, sistema inteligente de classificação de condutores, comportamento dos motoristas, desenvolvimento do projeto.

1 PROCESSADOR ARM

Os processadores ARM (*Advanced RISC Machine*) tem como principal característica, um baixo consumo de energia e um alto nível de processamento, essas características são desejáveis na maioria dos equipamentos eletrônicos. A arquitetura dos processadores ARM é RISC (*Reduced Instruction Set Computer*), uma comparação entre as arquiteturas RISC e CISC será apresentada no tópico 1.1. O núcleo ARM7 está disponível em vários microprocessadores de diversos fabricantes como: Philips, *Analog Devices*, OKI, entre outros (SOUZA, 2006).

1.1 Processadores CISC e RISC

Comparar as arquiteturas CISC (*Complex Instruction Set Computer*) e RISC (*Reduced Instruction Set Computer*) simplesmente como características para definir a melhor delas pode ser um grande equívoco. Para se realizar uma análise efetiva é necessário compreender o contexto histórico em que os fatos ocorreram. Devem-se entender essas duas arquiteturas como projetos distintos na busca de soluções para determinados conjuntos de problemas com suas tecnologias e estratégias em certo momento histórico (SILVA; ANTUNES 2007).

As principais características da arquitetura CISC são: conjunto de instruções complexas, sendo que uma instrução pode levar vários ciclos de relógio para ser resolvida; código fonte reduzido; *pipeline* com estrutura mais complexa e a decodificação das instruções, que podem levar mais de um ciclo (IBIDEM).

A arquitetura RISC tem como principais características: conjunto de instruções reduzidas; compilador das CPU´s RISC mais robusto, devido ao número reduzido de instruções, onde as mesmas são mais simples; código fonte mais longo pelo fato da CPU não possuir todas as instruções prontas (IBIDEM).

De forma prática nos processadores CISC, os programadores dispõem de quase todas as instruções que serão usadas nos programas, implementadas no processador, facilitando e reduzindo o tempo de desenvolvimento. Em relação a processadores estritamente RISC, os programadores tem mais trabalho, uma vez que dispõem apenas de instruções simples e

sempre que houver a necessidade de instruções complexas é necessário combinar várias instruções (MORIMOTO, 2001).

As descrições das arquiteturas RISC e CISC não descrevem mais as arquiteturas atuais. Atualmente os fabricantes estão na busca de uma alternativa que garanta vantagens sobre os seus concorrentes. As duas arquiteturas resistem no mercado pelas distintas vantagens, RISC pela *performance* e CISC pela compatibilidade de softwares (SILVA, ANTUNES 2007).

1.2 História do processador ARM

Em 1983 a *Acorn Computer Ltd* desenvolveu o primeiro modelo do processador ARM. Já em 1985 incidiu o lançamento do processador ARM1 repercutindo em produto, o sucesso foi tamanho que no ano adjacente foi lançado o ARM2. Em 1990 foi realizada uma parceria entre *Apple Computers*, VLSI *Technology* e a *Acorn Computer Ltd*, dessa parceria passou a existir a *Advanced RISC Machine Ltd* (RENALDI 2006).

A Advanced RISC Machine Ltd. não perfaz os processadores ARM, ela os projeta e licencia. Várias empresas de semicondutores produzem os processadores ARM adicionando suas próprias funcionalidades e periféricos. Algumas das empresas que produzem processadores ARM são: Philips, Atmel, Texas Instruments, Cirrus Logic, Intel, IBM, Sharp e Samsung, entre outras (IBIDEM).

Os processadores ARM podem ser divididos em famílias, sendo elas:

- **Processadores para aplicações**: Processadores projetados para executarem sistemas operacionais para aplicações embarcadas, uma aplicação dessa família seriam os telefones celulares. Podem ser citados os seguintes processadores que fazem parte dessa família: ARM1020E, ARM1022E, ARM1026EJS, ARM11 MPCore, ARM1136J(F)-S, ARM1176JZ(F)-S, ARM720T, ARM920T, ARM922T, ARM926EJ-S, entre outros;
- **Processadores para sistemas embarcados**: Processadores de soluções de tempo real, aplicações industriais e automotivas podem ser um exemplo do emprego dessa família de processadores. Os seguintes processadores que fazem parte dessa família: ARM Cortex-M3,

ARM1026EJS, ARM1156T2(F)-S, ARM7EJ-S, ARM7TDMI, ARM7TDMI-S, ARM946ES, ARM966E-S, ARM968E-S, entre outros;

- **Processadores *secure cores***: Processadores destinados a aplicações que necessitem de segurança, o seu núcleo incorpora várias características de segurança, esses processadores são utilizados em aplicações bancárias, TV paga, redes e biometrias. Alguns dos processadores que fazem parte dessa família: SecurCore SC100, SecurCore SC110, SecurCore SC200, SecurCore SC210.

1.3 Núcleo do processador ARM7

Segundo Sousa (2006) o LPC213x utiliza o núcleo ARM7TDMI-S. As principais características desse núcleo são:

- **Modo *Thumb*:** segundo conjunto de instruções com 16 bits;
- **Multiplicação longa:** capacidade de realizar multiplicações de 32 bits com resultados de 64 bits;
- **Depuração:** possui com extensão de hardware para depuração através da porta JTAG;
- ***Embedded* ICE:** extensão do módulo de depuração permite adicionar pontos de parada (*breakpoints*) e visualização dos registros através da porta JTAG;
- Alta capacidade de processamento: 0.9 MIPS/MHz;

O processador ARM possui dois conjuntos de instruções, ARM e *Thumb*. Quando a intenção do uso do processador ARM for velocidade, deve-se utilizar o conjunto de instruções ARM, esse modo garante uma *performance* em torno de 40% em relação ao modo *Thumb*, porém, quando se almeja economia de memória, o conjunto de instruções *Thumb* nos garante 30% de economia em relação ao conjunto de instruções ARM (SOUSA, 2006).

A Figura 1 apresenta o diagrama de blocos do núcleo do processador ARM7, esse diagrama foi retirado do manual técnico do ARM7. Fazendo uma breve análise sobre o diagrama de blocos apresentado, é possível reparar nos seguintes detalhes: multiplicador 32x8, unidade lógica aritmética, contador de programa, decodificador de instruções, decodificador de instruções *Thumb*, controle de *debug*, dentre outros (ARM, 2004).

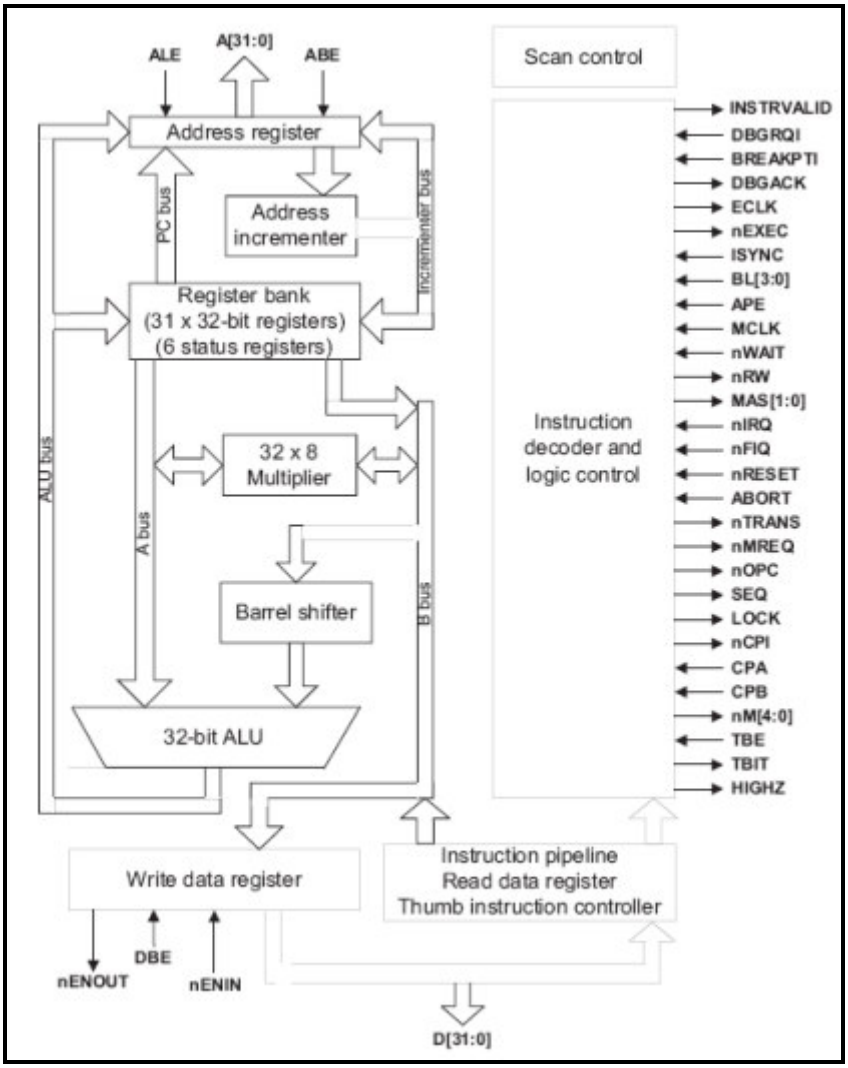

Figura 1 - Diagrama em blocos do núcleo ARM7TDMI-S
Fonte: Manual de referência do processador ARM.

O acesso a memória nos processadores ARM é estruturalmente Von Neumann, com barramento de 32 bits para acesso a memória do programa e dados ao mesmo tempo. Em arquiteturas RISC, o padrão de acesso a memória seria Harvard, que consiste em duas memórias separadas, uma memória para o programa e outra para os dados. Já na arquitetura Von Neumann existe uma única memória tanto para o programa quanto para os dados (SOUSA, 2006).

A estrutura de *pipeline* dos processadores ARM7 consiste em três estágios: busca, decodificação e execução, salientando que cada estágio é independente. Com o uso do *pipeline,* a maioria das instruções é executada em um ciclo de máquina, com exceção de

quando é dado o início no ciclo. Na Figura 2 é possível visualizar o ciclo das tarefas do *pipeline* (IBIDEM).

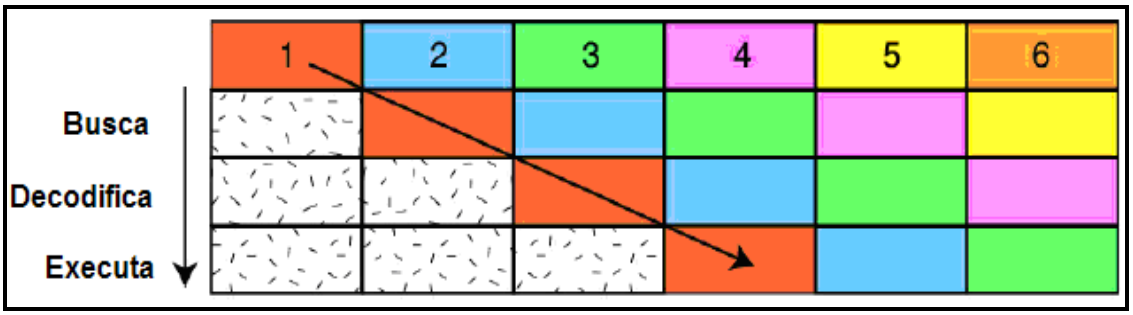

Figura 2 – Estrutura de *pipeline* do processador ARM
Fonte: Livro de Daniel Rodrigues SOUSA - Microcontroladores ARM7

Todas as operações realizadas no processador ARM7 fazem o uso de registradores. Isso deve-se ao fato de o processador possuir uma estrutura *load-and-store*. Na Figura 3 pode ser visualizado um pequeno exemplo de soma. A idéia nesse exemplo é a de realizar a soma de M1 com M2, e que o resultado esteja na variável M3. Para este fim, faz-se necessário mover o valor de M1 para o registrador R1, e o valor de M2 para R2. Na seqüência é feita a operação de soma que mantém o valor no registrador quatro. Para acessar o resultado é necessário mover o valor do R4 para a variável M3 (IBIDEM).

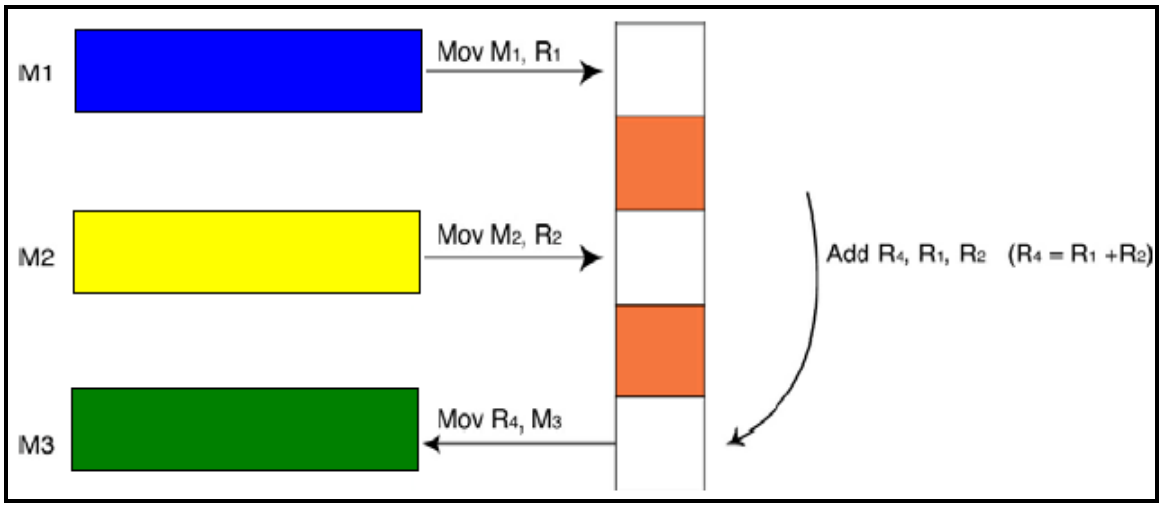

Figura 3 - Exemplo do uso de registradores
Fonte: *LPC ARM Book*

Na Figura 4 é possível visualizar os sete modos de operação do processador ARM7. A diferença entre os modos está no uso dos registradores. Os sete modos de operação do processador são:

- **Usuário**: Execução normal de programas;
- **Sistema**: executa rotinas privilegiadas do Sistema Operacional;
- **Supervisor**: Modo protegido para o Sistema Operacional;
- **IRQ**: Tratamento de interrupções comuns;
- **FIQ**: Tratamento de interrupções rápidas;
- ***Abort***: Usado para implementar memória virtual ou proteção de memória;
- **Indefinido**: Suporta a emulação em software de co-processadores.

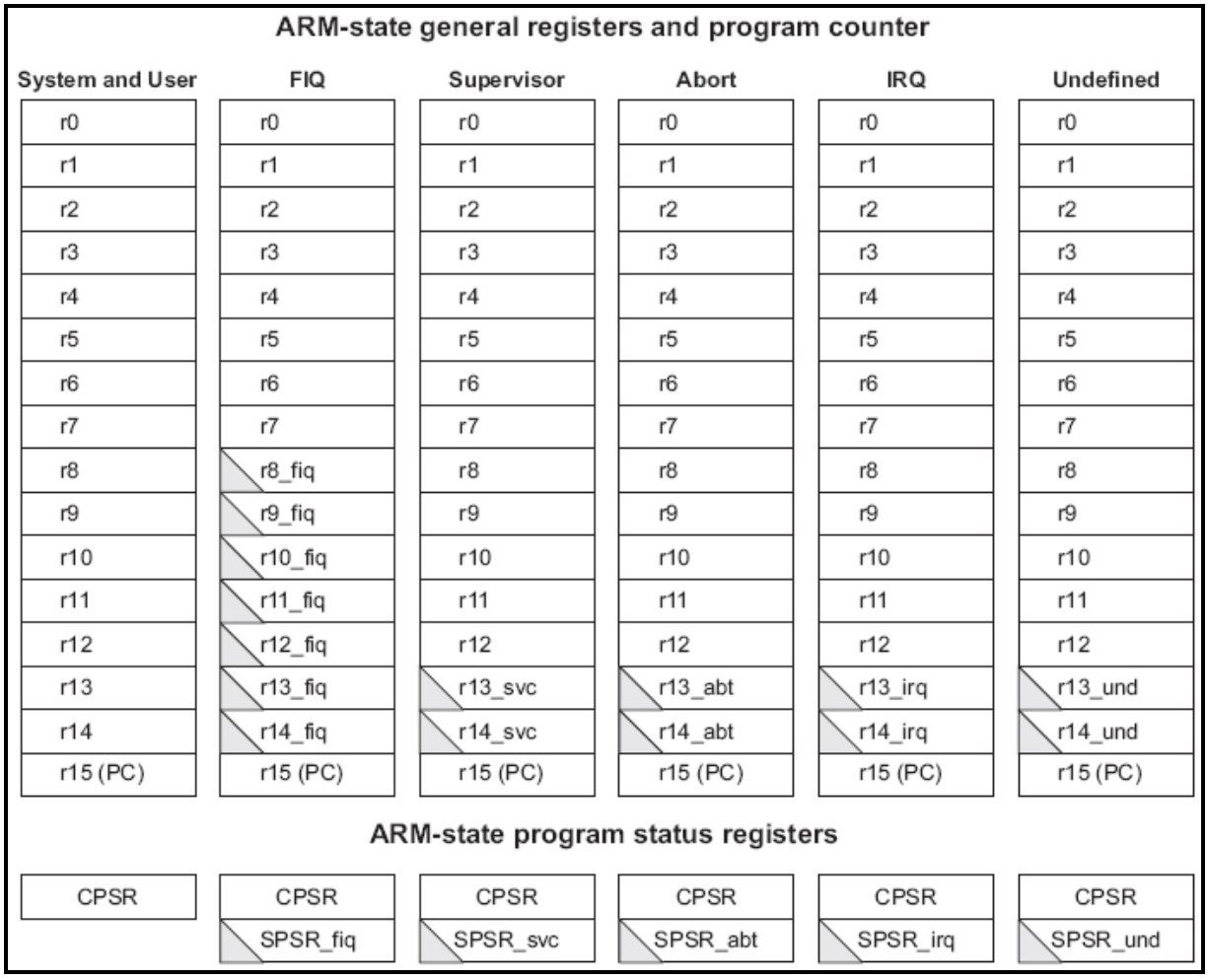

Figura 4 – Os sete modos de operação do processador ARM7
Fonte: Manual de referência do processador ARM.

O núcleo do processador ARM7 possui dois modos de trabalho das variáveis: *big-endian* e *little-endian*. No modo *big-endian,* o byte menos significativo está na esquerda, já no modo *little-endian,* o byte menos significativo está na direita. O padrão da Philips é *little-endian*, mais comumente encontrado na maioria dos microprocessadores e microcontroladores. A Figura 5 ilustra a diferença entre os dois modos, repare que a demonstração realizada na figura leva em conta um valor de quatro bytes que representam 32 bits (0-31), no modo *little-endian* o byte A está na direita e no modo *big-endian* o byte A está na esquerda (ARM, 2004).

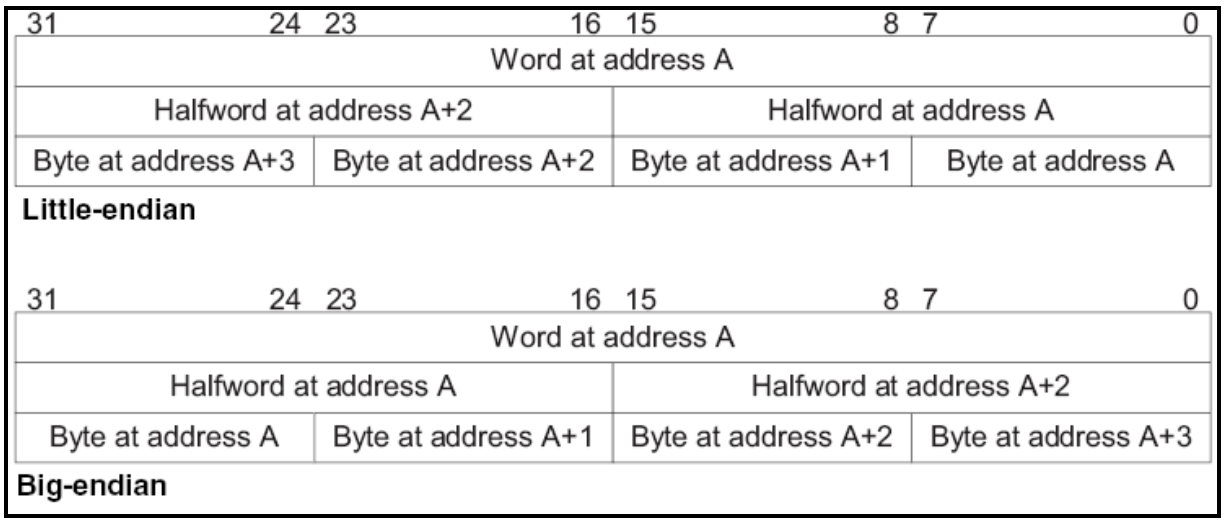
31 24 23 16 15 8 7 0
Word at address A
Halfword at address A+2
Halfword at address A
Byte at address A+3
Byte at address A+2
Byte at address A+1
Byte at address A
Little-endian
31 24 23 16 15 8 7 0
Word at address A
Halfword at address A
Halfword at address A+2
Byte at address A
Byte at address A+1
Byte at address A+2
Byte at address A+3
Big-endian

Figura 5 - Duas formas de tratamento das variáveis. *Little-endian*/ *Big-endian*
Fonte: Manual de referência do processador ARM.

1.4 Características do Processador ARM7 LPC213x

Conforme Sousa (2006), a família de processadores LPC213x é baseada no núcleo ARM7TDMI-S. O seu tamanho, consumo e desempenho são ideais para as mais diversas aplicações como: equipamentos médicos, controladores industriais de uso geral, controle de acesso, conectividade e aplicações de uso geral. A aplicação proposta por esse trabalho utilizará o processador LPC2138 da Philips e as principais características desse processador são:

- Núcleo ARM7TDMI-S;
- Alimentação de 3,0V a 3,6V;

- 512 Kbytes de memória de programa;
- 32 Kbytes de memória de dados volátil (RAM);
- 22 interrupções (quatro externas);
- 47 I/Os;
- Dois *timers* ou contadores de eventos externos de 32 bits;
- Unidade de PWM com seis saídas;
- Watchdog Timer;
- Duas UART`S;
- Dois barramentos I²C`S;
- Um barramento SPI;
- Um módulo SSP;
- RTC interno;
- Dois conversores A/D de 10 bits com oito canais cada;
- Um canal de conversão D/A de 10 bits;
- Opera com cristal de 1 MHz até 30 MHz ou oscilador externo de 1 MHz até 50 MHz;
- 60 MHz de operação máxima via PPL interno.

1.5 Pinagem do processador ARM7 LPC2134/ 2136/ 2138

O processador ARM é do tipo SMD (*Superficial Monting Device*), significando que este fica sobre a placa de circuitos impressos, ao contrário dos processadores antigos que eram fixados no lado oposto da placa. Esse detalhe permite que os processadores SMD possam ser menores, porém a sua troca pode ser muito mais trabalhosa.

A Figura 6 demonstra a distribuição dos pinos do processador ARM7 LPC2138. Através da imagem, esclarece-se que o processador possui 64 pinos e cada um destina-se a uma função. Algo que a princípio parece simplista, passando muitas vezes despercebido, é o marcador de posição, reparem que no canto superior esquerdo existe uma pequena circunferência que indica o local do pino um (NXP, 2008).

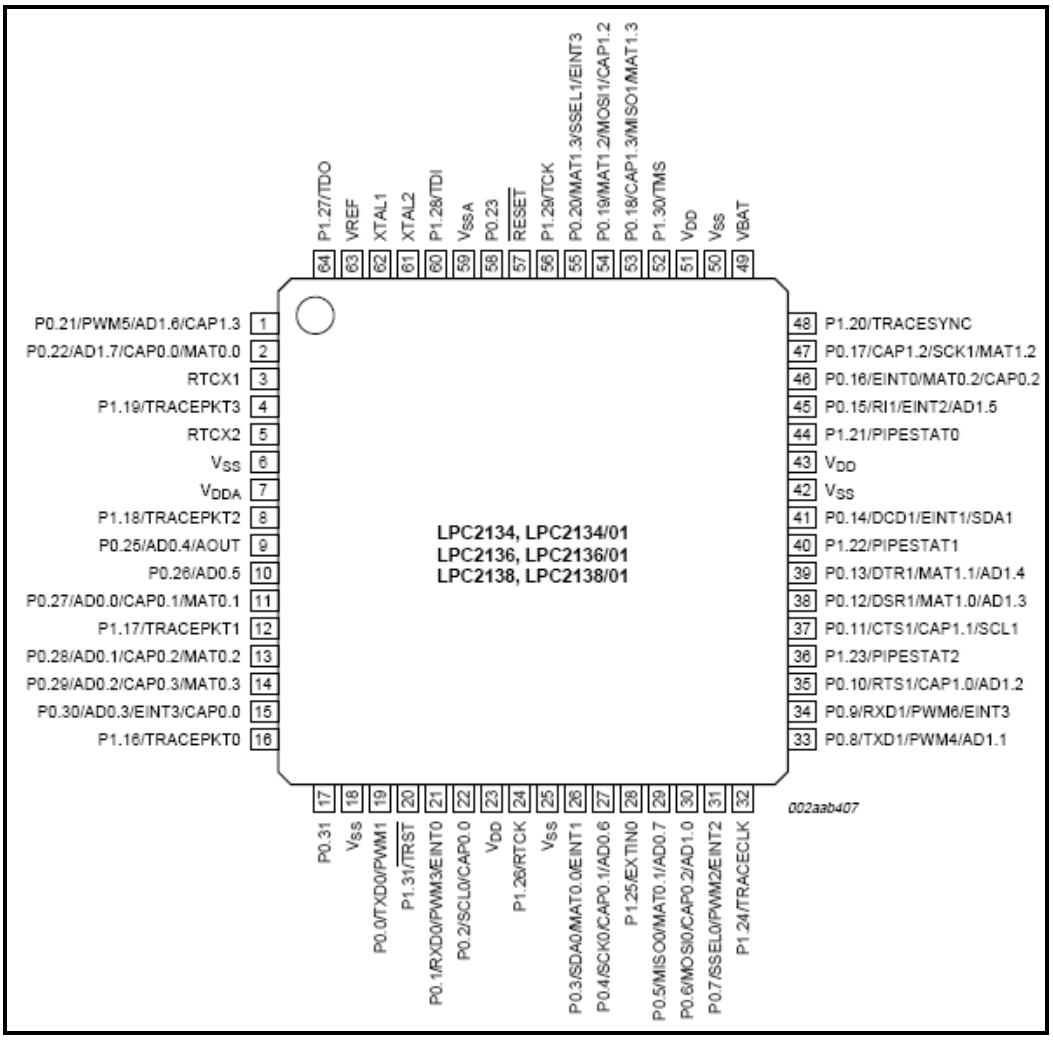

Figura 6 – Distribuição dos pinos do processador ARM7 LPC2138
Fonte: Manual de referência do processador ARM.

1.6 Barramentos do processador ARM7 LPC2138

Na arquitetura do processador ARM7 existem três barramentos: ARM7 *local bus* (dedicado à comunicação do núcleo com as memórias SRAM e Flash), AHB (*Advanced High Performance Bus,* idealizado para comunicação mais rápida entre o núcleo e os periféricos, ligado ao vetor de controle de interrupções e a ponte AHB/ APB) e APB (ARM *Peripheral Bus*, todos os periféricos estão ligados a ele.). Na Figura 7 pode ser visualizado o esquema dos barramentos (ARM, 2004).

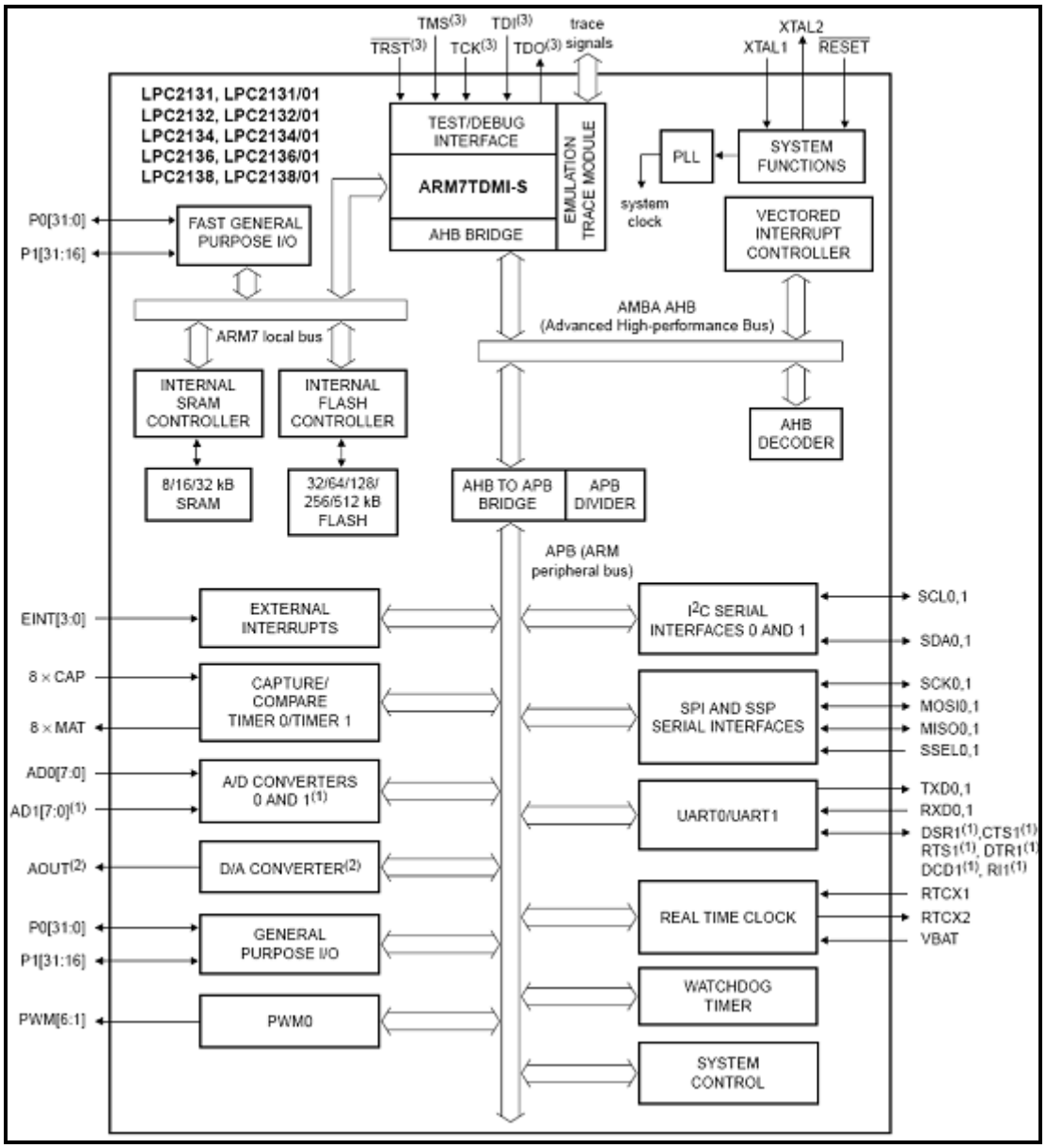

Figura 7 - Sistema de barramentos ARM7
Fonte: Manual de referência do processador ARM.

1.7 Mapa de memória do processador ARM7

Na Figura 8 é possível visualizar o esquema de organização de memória do processador ARM7. Para entender esse mapeamento é importante ressaltar que a memória pode ser: Flash (*NON-VOLATILE MEMORY*) ou SRAM (*Static RAM*). Analisando a imagem de baixo para cima, é possível observar que, conforme o modelo do processador, a memória é maior. Tem-se como exemplo a capacidade do modelo LPC2131, que possui uma capacidade de 32 Kbytes e o modelo LPC2138 é de 512 Kbytes. Acima da memória *flash* estão definidos

os endereços da memória SRAM, também organizada pelos modelos de processador, e por último temos os endereços destinados aos periféricos e as interrupções (NXP, 2008).

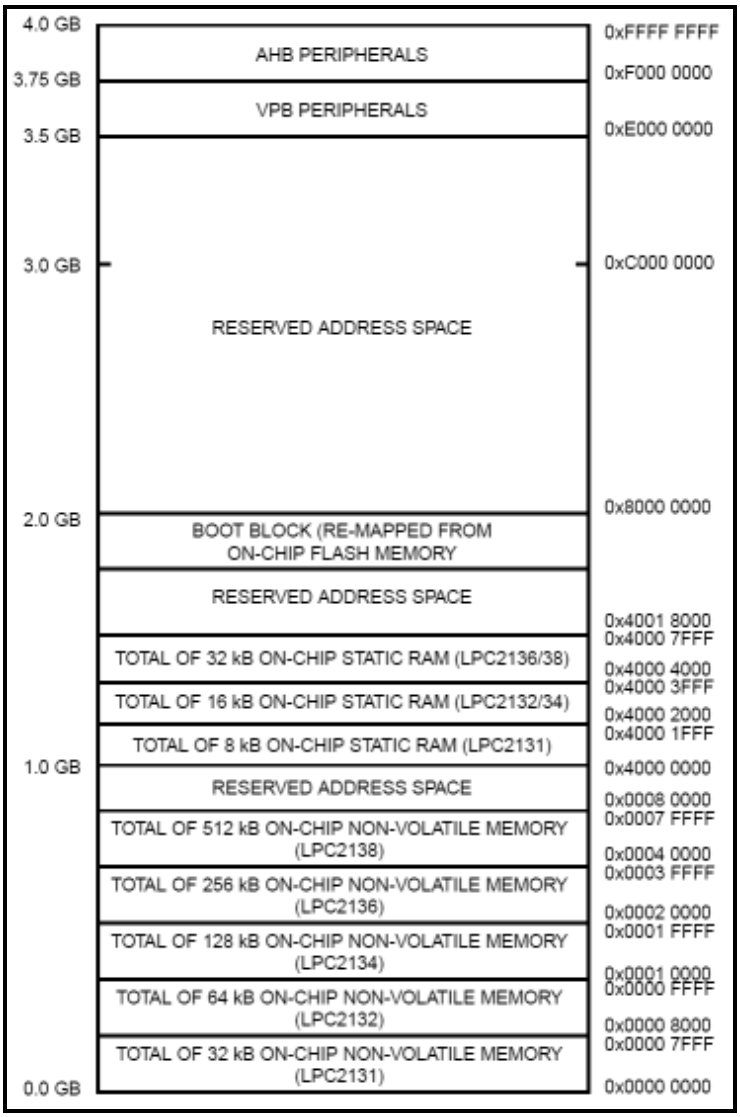

Figura 8 - Mapa da memória processador ARM.
Fonte: Manual Philips LPC2131/32/34/36/38

1.8 Sistema de interrupções do processador ARM7 LPC2138

Para gerenciar e controlar as interrupções, existe um módulo chamado VIC (*Vectored Interrupt Controller*), nesse módulo estão implementadas tanto as interrupções externas, quanto as de periféricos. A prioridade das interrupções também é determinada por linha de código no módulo VIC que está ligado através do barramento AHB ao núcleo do processador ARM7 (SOUSA, 2006).

No VIC estão localizadas todas as interrupções classificadas como: FIQ (*Fast Interrupt Request*), IRQ (*Vectored Interrupt Request*) e *non-vectored* IRQ. A interrupção FIQ tem maior prioridade, as interrupções IRQ tem prioridade média e as interrupções *non-vectored* IRQ tem a menor prioridade (NXP, 2008).

O núcleo do processador reserva uma parte dos registradores para tratamento das interrupções. Conforme o tipo de interrupção a ser tratada, é reservado um conjunto específico de registradores como forma de otimizar o tratamento das interrupções (SOUSA, 2006).

O tempo de atendimento de uma interrupção FIQ está na ordem de 200ns com o processador atuando em 60MHz, em contrapartida, o atendimento a uma interrupção IRQ está na casa de 416ns com o processador a 60 MHz (IBIDEM).

A Tabela 1 exibe a lista das principais interrupções e os seus respectivos canais, sendo extremamente importante para o desenvolvimento dos sistemas, pois para habilitar uma interrupção é necessário conhecer o seu canal. Um ensaio utilizando interrupção pode ser visto no ANEXO A.

Tabela 1 - Lista das principais interrupções

BLOCK	VCI *Channel*
WDT	0
ARM Core	2
ARM Core	3
Timer 0	4
Timer 1	5
UART 0	6
UART 1	7
PWM0	8
EINT 0/1/2/3	14/ 15/ 16/ 17

Fonte: Manual de referência NXP.

1.9 Comparativo técnico entre alguns processadores ARM

Tendo em vista que o processador não é o item mais significante desse projeto e sim a inteligência para a resolução do objetivo principal, abaixo segue uma tabela com as principais características dos principais processadores ARM fabricados pela Philips.

A Tabela 2 tem o objetivo de demonstrar o grande número de microcontroladores existentes no mercado. Foram listados apenas alguns processadores ARM produzidos pela Philips.

Tabela 2 - Comparativo entre alguns processadores ARM fabricados pela Philips

Modelo	CPU Clock	Mem. Flash	RAM	Temperatura
ARM7 / LPC2104BBD48	60 MHz	128 KB	16 KB	0 °C to +70 °C
ARM7 / LPC2105FBD48/00	60 MHz	128 KB	32 KB	0 °C to +70 °C
ARM7 / LPC2106FHN48	60 MHz	128 KB	64 KB	-40 °C to +85 °C
ARM7 / LPC2106FHN48/00	60 MHz	128 KB	64 KB	-40 °C to +85 °C
ARM7 / LPC2131FBD64/01	60 MHz	32 KB	8 KB	-40 °C to +85 °C
ARM7 / LPC2132FBD64/01	60 MHz	64 KB	16 KB	-40 °C to +85 °C
ARM7 / LPC2132FHN64	60 MHz	64 KB	16 KB	-40 °C to +85 °C
ARM7 / LPC2134FBD64	60 MHz	128 KB	16 KB	-40 °C to +85 °C
ARM7 / LPC2136FBD64	60 MHz	256 KB	32 KB	-40 °C to +85 °C
ARM7 / LPC2138FBD64	60 MHz	512 KB	32 KB	-40 °C to +85 °C
ARM9 / LH7A400N0F076B5	250 MHz		80 KB	-40 °C to +85 °C
ARM9 / LH7A400N0F000B3A	200 MHz		80 KB	-40 °C to +85 °C

Fonte: www.nxp.com, página oficial da Philips

O mais importante no momento de desenvolver um projeto embarcado é escolher um processador pelo custo/benefício. A exemplo disso cita-se a aplicação embarcada que sempre estará em temperatura entre 15 °C a 40 °C. Utilizar um processador que suporte uma variação grande de temperatura é um desperdício. O exemplo anterior vale para qualquer comparação do gênero, levando outras características em conta, como: *clock*, memória flash, memória RAM (NXP, 2008). O estudo do processador utilizado neste projeto é fundamental, pois, para desenvolver o projeto é necessário um conhecimento aprofundado das funcionalidades do mesmo.

2 SOFTWARE DE DESENVOLVIMENTO

O software analisado para o desenvolvimento foi o IDE *MicroVisio*n. Existem demais alternativas no mercado, entretanto, essa acompanhava o kit de desenvolvimento utilizado. Keil Software, fabricante da IDE *MicroVisio*n é uma empresa que visa o desenvolvimento de aplicações embarcadas para clientes do ramo eletrônico, indústrias, comunicação, automação e automotivas (KEIL, 2008).

A Keil Software desenvolveu a IDE *MicroVisio*n, que permite o desenvolvimento de aplicações embarcadas em C e *Assembly* no mesmo código fonte. Essa IDE será abordada com mais detalhes posteriormente. A versão da IDE *MicroVision* 3 será utilizada no desenvolvimento do projeto corrente.

Essa empresa, Keil Softvare, também desenvolve e comercializa kits que possuem como ponto central e mais relevante, um processador, seja esse da família ARMx ou x86 e ligado a esse processador uma variedade de I/Os como: LED´s, botões, som, portas seriais, *displays*, JTAG. A maioria dos kits de desenvolvimento está preparada para que o usuário possa adicionar outras I/Os na placa sem contar com a possibilidade de adicionar outros equipamentos que podem ser acoplados nesse hardware. A placa MCB2130 é o hardware a ser utilizado nesse trabalho, onde tem-se uma visão mais aprofundada sobre ela.

2.1 Kit de desenvolvimento MCB2130

O kit de desenvolvimento MCB2130 da Keil é adequado para realizar protótipos de software sobre o processador ARM7. Para utilização da placa de desenvolvimento é necessário possuir um dos seguintes sistemas operacionais: Microsoft Windows XP, Microsoft Windows 2000 ou Microsoft Windows Vista. Será necessária uma porta USB para a alimentação do circuito e uma porta serial para a transferência do código HEXA para o processador ARM utilizando o aplicativo *Flash Utility* LPC2000 desenvolvido pela Philips. Na Figura 9 é possível visualizar o hardware de desenvolvimento projetado e desenvolvido pela Keil (KEIL, 2008).

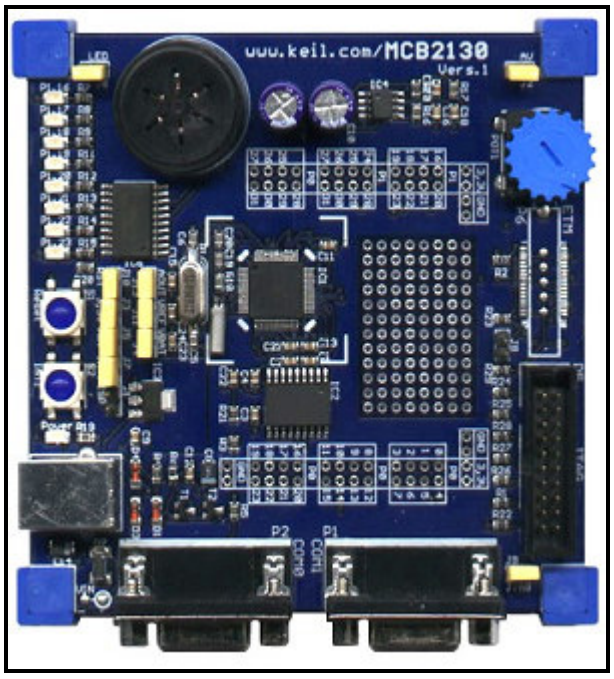

Figura 9 - Kit de desenvolvimento MCB2130
Fonte: site www.keil.com

A configuração da MCB2130 é realizada através de *jumpers* localizados próximos ao botão de *start* e o botão de *reset*. Um exemplo da utilização dos *jumpers* seria a configuração da saída dos LED`s, se o *jumper* 06 (J06) estiver habilitado então será possível acionar a saída dos LED`s, caso contrário, não será possível acionar a saída dos LED`s.

A Tabela 3, específica as principais configurações realizadas através de *jumpers* na placa de desenvolvimento MCB2130 (KEIL, 2004):

Tabela 3 - Configuração da placa MCB2130 através de *jumpers*

Jumper	Estado	Descrição
J01- ISP	ON	Habilitar a programação através da porta COM0
J02- AV	ON	Conecta potenciômetro POT1na entrada analógica AIN0.
J03- 3.3V	ON	Conecta 3.3V to CPU.
J04- VBAT	ON	Conecta VBAT to CPU.
J05- V3A	ON	Conecta voltagem analógica na CPU.
J06- LED	ON	Habilita Porta 1.16 – Porta 1.23 dos *LED's*.
J07- INT1	ON	Habilita o botão INT1.
J08- ETM	OFF	Desabilita e habilita *trace*.
J09- JTAG	ON	Habilita a interface *JTAG*.
J10- RST	ON	Habilita *Reset* pela COM0.
J11- VREF	ON	Conecta VREF na CPU.
J13- AOUT	ON	Conecta saída amplificador na saída de som.

Fonte: Tabela retirada do manual do fabricante da placa MCB2130.

Além do conjunto de I/Os disponíveis, o kit da Keil possui uma arquitetura preparada para receber outros dispositivos de entrada e saída, como: LED`s, displays, botões além de dispositivos que podem ser acoplados a esse hardware (KEIL, 2008).

Na placa de desenvolvimento MCB2130, no canto superior direito existe uma entrada para um componente chamado JTAG, através dessa interface é possível encontrar prováveis erros no programa em alto nível de software. A interface JTAG permite interligar o kit de desenvolvimento com a IDE de programação, possibilitando verificar os valores das variáveis em tempo de execução do software (IBIDEM).

2.2 Hardware MCB2130

A Figura 10 exibe um diagrama de bloco da estrutura do kit MCB2130 de desenvolvimento. Através desse diagrama fica clara a compreensão das I/Os que estão ligadas ao processador ARM LPC2138. É possível notar que as setas indicam o sentido dos dados, como exemplo: porta *RS232* tem comunicação de entrada e saída, a porta dos LED`s tem apenas sentido de saída e a alimentação de energia tem função de entrada apenas.

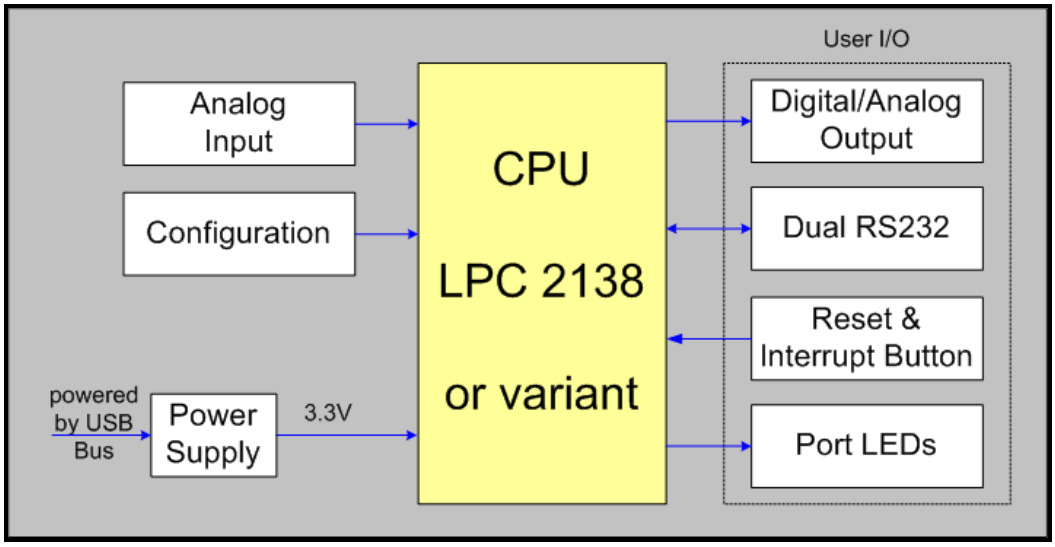

Figura 10 - Diagrama de blocos do kit de desenvolvimento MCB2130
Fonte: site www.keil.com

As principais características técnicas encontradas no kit MCB2130 são:

- MCU LPC2138;
- MCU Clock 60 MHz;
- XTAL 12 MHz;
- Processador ARM ARM7TDMI-S;
- Área Prototipagem;
- JTAG Interface;
- Dimensões 10cm X 10cm;
- On-Chip RAM 32K;
- On-Chip FLASH 512K;
- Dois botões;
- Oito LED`s;
- Potenciômetro;
- Alto-falante;
- Duas portas seriais;
- Alimentação pela porta USB.

Um dos pontos fortes dos processadores ARM7 é o baixo consumo de energia e isso permitiu a Keil desenvolver uma placa com um consumo de energia baixo, podendo ser alimentada pela porta USB. A energia necessária é de 3,3 V. Esse fator é importante, devido ao fato de muitas aplicações dependerem do uso de baterias e assim não disponibilizarem de uma grande quantidade de energia. Um exemplo clássico são os celulares, onde grande parte dos celulares da Nokia utiliza processador ARM e como requisito fundamental tem-se o consumo de bateria, quanto maior a duração melhor (SILVESTRE; BACHIEGA, 2007).

2.3 Ferramenta de desenvolvimento *MicroVision* 3

MicroVision 3 é a interface de desenvolvimento e compilador oferecido pela Keil Software, também fabricante do kit de desenvolvimento MCB2130, através dessa ferramenta é possível criar os projetos, subdividir os projetos em unidades e gerar o código HEXA para ser usado no processador ARM. A IDE foi desenvolvida para ser executada sobre o sistema

operacional Windows e conta com um compilador C e *macro assembler* (KEIL, 2008). Nessa ferramenta é necessário configurar o hardware que está sendo utilizado, assim como as variáveis de compilação, enquanto que o restante dos seus itens se assemelham a outras ferramentas gráficas de desenvolvimento.

Visualizando a Figura 11, que ilustra a IDE *MicroVison,* é possível observar a sua divisão gráfica de tarefas. Na parte superior encontra-se o menu principal e logo abaixo é possível ter acesso aos botões de acesso rápido. No lado esquerdo tem-se como principal função manter aos olhos do programador a estrutura e organização do seu projeto. A parte central e maior é destinada a escrita do código fonte. Esse quesito possui vários recursos como: Minimizar e Maximizar as funções, adicionar pontos de parada do código fonte nas simulações, linhas de erros assinaladas.

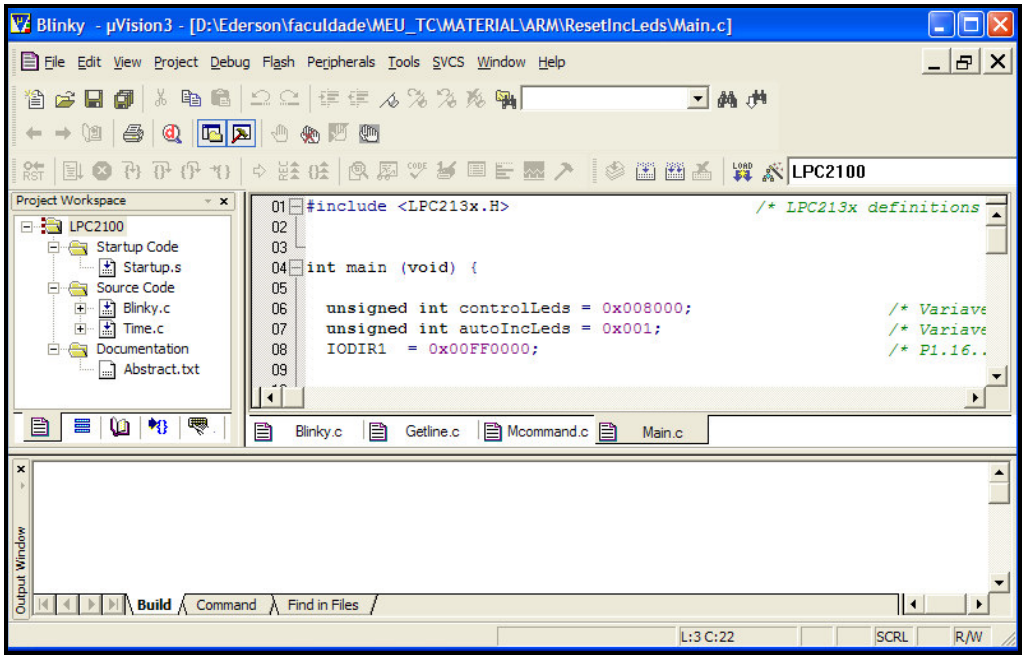

Figura 11 - IDE de desenvolvimento *MicroVision* da Keil Software
Fonte: do autor

Ao iniciar um projeto com a IDE *MicroVison* deve-se fazer alguns ajustes nas opções do projeto, sendo elas as seguintes:

- Definir o processador utilizado pelo projeto, isso se deve ao fato da IDE suportar um grande número de processadores;
- Habilitar como saída a criação do código fonte em HEXA;

- Vincular a IDE o software de gravação da memória flash;

Todos os itens acima podem ser visualizados na Figura 12.

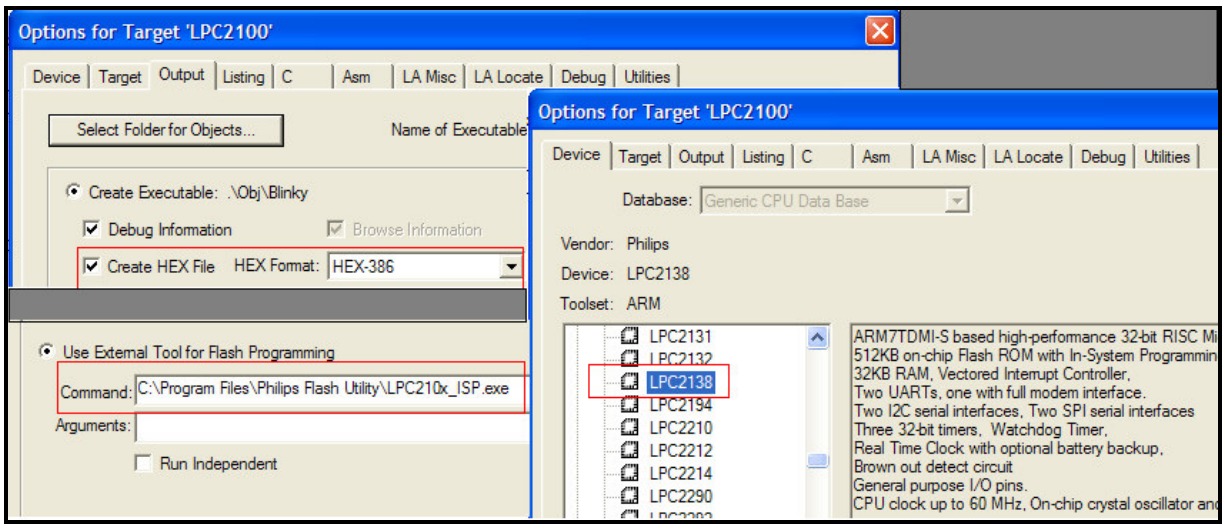

Figura 12 - Principais configurações de um projeto utilizando a IDE *MicroVision*
Fonte: do autor

Conhecer a IDE de desenvolvimento, garante um processo de desenvolvimento mais ágil. Para conseguir o dinamismo necessário fez-se um estudo avançado da ferramenta *MicroVison,* desenvolvida pela Keil Software. Porém, é necessário estudar softwares de gravação do processador, pois a IDE estudada não possui este recurso. No próximo capítulo serão estudados dois softwares capazes de realizar essa tarefa.

3 APLICATIVOS DE GRAVAÇÃO DA MEMÓRIA FLASH

Este capítulo será utilizado para apresentar o procedimento de atualização da memória *flash* do processador ARM e os aplicativos capazes de realizarem essa tarefa, entre os analisados encontram-se: *Philips Flash Utility* LPC2000 e o Flash Magic.

3.1 Aplicativo *Philips Flash Utility* LPC2000

O aplicativo *Philips Flash Utility* LPC2000 foi desenvolvido pela *Philips Semiconductors* em 2003 e tem os seus direitos reservados. A sua principal função é realizar a gravação do código HEXA no processador. Esse utilitário compreende uma grande gama de processadores e entre eles está o processador a ser utilizado nesse projeto LPC2138.

Para utilizar o aplicativo *Flash Utility* LPC2000 é necessário ter gerado anteriormente o código fonte HEXA, isso pode ser feito através do *MicroVision* 3 da Keil Software ou de qualquer compilador compatível. De posse do código em HEXA gerado, basta realizar algumas configurações no software para fazer a gravação da memória flash do processador. Os principais parâmetros a serem configurados são:

- *Filename*: Localização do código fonte em HEXA;
- *Device* : Selecionar o Processador a ser utilizado;
- *XTAL Freq*: Definir a freqüência do oscilador;
- *Connected to Port*: Definir a porta serial do PC que vai estar ligado ao hardware do processador;
- *Use Baud Rate*: Definir a velocidade de comunicação através da porta serial do PC.

Após a realização das principais configurações, basta simplesmente pressionar sobre o botão *Upload to Flash* para que a memória *flash* do processador seja gravada, deve-se lembrar que, talvez possa existir alguma configuração no hardware, no caso da placa de desenvolvimento da Keil Software é necessário realizar uma configuração nos *jumpers* para que ela permita a gravação na memória *flash*. Todos os detalhes descritos até o momento sobre o aplicativo *Flash Utility* LPC2000 podem ser observados na Figura 13.

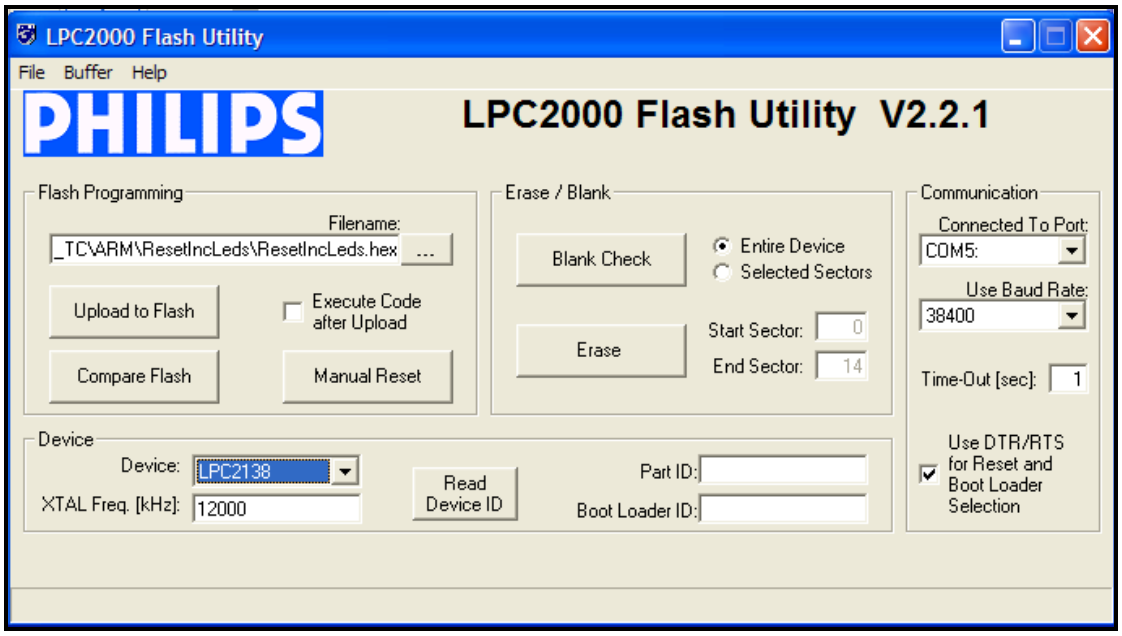

Figura 13 – Aplicativo *Flash Utility* LPC2000 desenvolvido pela *Philips Semiconductors*
Fonte: do autor

3.2 Aplicativo *Flash Magic*

O aplicativo *Flash Magic* é um sistema gratuito destinado a realizar a gravação da memória *flash* de microcontroladores da linha NXP, desenvolvido para ser executado sobre o sistema operacional da Microsoft® com exceção a versões anteriores ao Windows 98, necessita de apenas 10MB de espaço em disco rígido para ser instalado. Entre os dispositivos suportados pelo *Flash Magic* estão as seguintes séries: P89xxxxx, LPC21xx, LPC22xx, LPC23xx, PXA-Gxx (FLASH MAGIC, 2008).

Na Figura 14 é possível visualizar o software *Flash Magic*, entre as suas principais características pode-se frisar as seguintes: visualmente existe uma separação entre os principais passos a serem executados para realizar a gravação da memória *flash* do dispositivo almejado. O primeiro passo contempla a configuração da comunicação serial, onde deve-se selecionar a porta serial e a taxa de comunicação a serem utilizados, na seqüência deve ser informado o dispositivo utilizado e por último a freqüência do oscilador. O segundo passo é destinado a questão de apagar a memória *flash* antes de iniciar a gravação do novo código hexa. O terceiro passo é o mais simples de todos, é utilizado para localizar o código hexa a ser gravado, sendo que o aplicativo em questão informa a data e hora da última modificação. O

quarto passo é destinado a verificar o código hexa gravado ao fim do processo de gravação. Após devidamente configurados os passos acima, basta executar a gravação no quinto passo (IBIDEM).

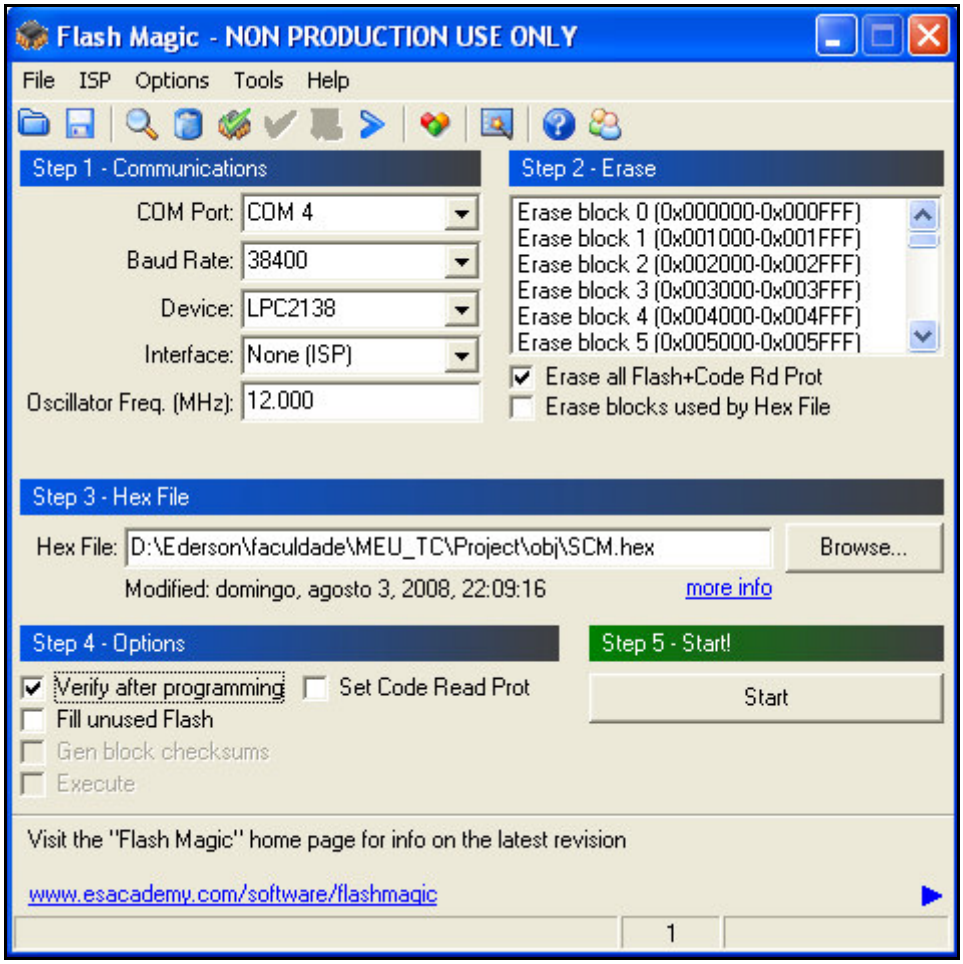

Figura 14 - Aplicativo Flash Magic
Fonte: do autor

O aplicativo *Flash Magic* traz consigo um manual completo sobre as suas funcionalidades e modos de operação, onde encontram-se informações para integrá-lo a IDE *MicroVision* 3 de desenvolvimento. Explorando o aplicativo pode-se encontrar várias características e funcionalidades importantes, como:

- Informações: esse item disponibiliza informações sobre o código hexa como a data da criação, data da última modificação, localização do arquivo, tamanho, percentual de utilização da memória *flash* do dispositivo e o CRC;
- *Blank Check*: utilizado para visualizar os blocos que estão em branco ou não;

- Assinatura do dispositivo: destinado a ler a assinatura do processador, sendo que o primeiro byte identifica o fabricante, no caso da NXP o valor em hexadecimal é 15;
- Exibir memória: essa tela exibe o conteúdo da memória *flash*, sendo necessária a escolha do bloco a ser exibido. É possível visualizar a memória no formato hexadecimal e ao lado no formato ASCII;
- *Eransing flash*: utilizado para apagar blocos da memória selecionados pelo usuário;
- Verificar o arquivo hexa: utilizado para verificar a integridade do arquivo hexa gravado no microcontrolador;
- Executado em linha de comando: essa funcionalidade permite o acionamento das funcionalidades do Flash Magic sem a Interface gráfica apresentada na Figura 14, para a utilização dessas funcionalidades basta configurar a pasta do aplicativo na variável de ambiente do Windows "PATH", os possíveis comandos podem ser encontrados no manual presente no aplicativo;
- Terminal: permite abrir um terminal de comunicação com o dispositivo, onde é possível enviar informações e visualizar os retornos;
- API: é possível utilizar a DLL do aplicativo Flash Magic para a implementação do próprio sistema de gravação de memória *flash*.

Devido a todas as características apresentadas sobre o Flash Magic, onde este apontou ser superior ao utilitário LPC2000 da Philips, ele será adotado como utilitário de escolha na implementação do projeto proposto pelo presente trabalho. Até o momento, foram apresentados os dois softwares de terceiros utilizados para desenvolver a aplicação embarcada, o *MicroVision* (IDE de desenvolvimento) e o *Flash Magic*. O uso de aplicações de terceiros é inevitável, sendo assim, é importante conhecer as ferramentas no mínimo a um nível básico para desenvolver a aplicação.

4 SISTEMA DE POSICIONAMENTO GLOBAL - GPS

O sistema *Global Position System* (GPS), é um sistema americano de posicionamento por satélite. Nas edições subseqüentes, serão estudados o histórico, funcionamento, receptores e o protocolo NMEA.

4.1 História do GPS

O GPS surgiu devido a corrida armamentista entre Estados Unidos (EUA) e a extinta União Soviética (URSS) para o reconhecimento de uma determinada posição em tempo real. Os soviéticos por outro lado desenvolveram o *Glonass* (*Global Navigation Satellite System*). Os cientistas norte-americanos observando as ondas de rádio do Sputnik I, lançado em 1957, puderam analisaram que através do efeito *doppler* era possível saber o posicionamento do satélite na órbita e em contrapartida era possível calcular a posição do receptor na terra (ROCHA, 2002).

Os americanos deram início ao projeto GPS em 1973, sendo ele inicialmente uma extensão do projeto NNSS/TRANSIT da Marinha Americana. O primeiro satélite GPS foi lançado em fevereiro de 1978 (IBIDEM).

Quando implementado o GPS, os projetistas esperavam um erro na ordem de 400 metros para usuários civis, porém testes realizados por receptores civis evidenciaram uma precisão na prática de 15 a 40 metros. Esse fato, sobre pretexto de segurança fez com que fosse imposto um distúrbio proposital no relógio dos satélites. Essa imprecisão foi eliminada no dia primeiro de maio de 2000, pelo presidente Norte-Americano Bill Clinton. A partir dessa data a precisão dos receptores civis é equivalente aos de uso militar (IBIDEM).

4.2 Funcionamento do GPS

Existem 24 satélites operacionais na órbita terrestre, sua altura em relação ao mar é de aproximadamente 20.200 km. Eles estão divididos em seis órbitas com uma inclinação de 55º

em relação ao Equador. Os satélites percorrem uma órbita completa a cada 12 horas e cada satélite tem 28º de visualização sobre a Terra. Todo ponto terrestre em qualquer instante recebe o sinal de pelo menos quatro satélites, porém é possível que em determinados momentos um receptor possa visualizar até dez satélites (ALVES, 2006).

O GPS é didaticamente dividido em três segmentos: Segmento Espacial (Satélites e seus sinais), Segmento de Controle (Responsável pela monitoração, geração e avaliação de todo o sistema) e Segmento de Usuários (todos os tipos de aplicações, métodos de posicionamento, formas de recepção, entre outros) (ROCHA, 2002).

A Figura 15 demonstra a disposição dos satélites na órbita terrestre.

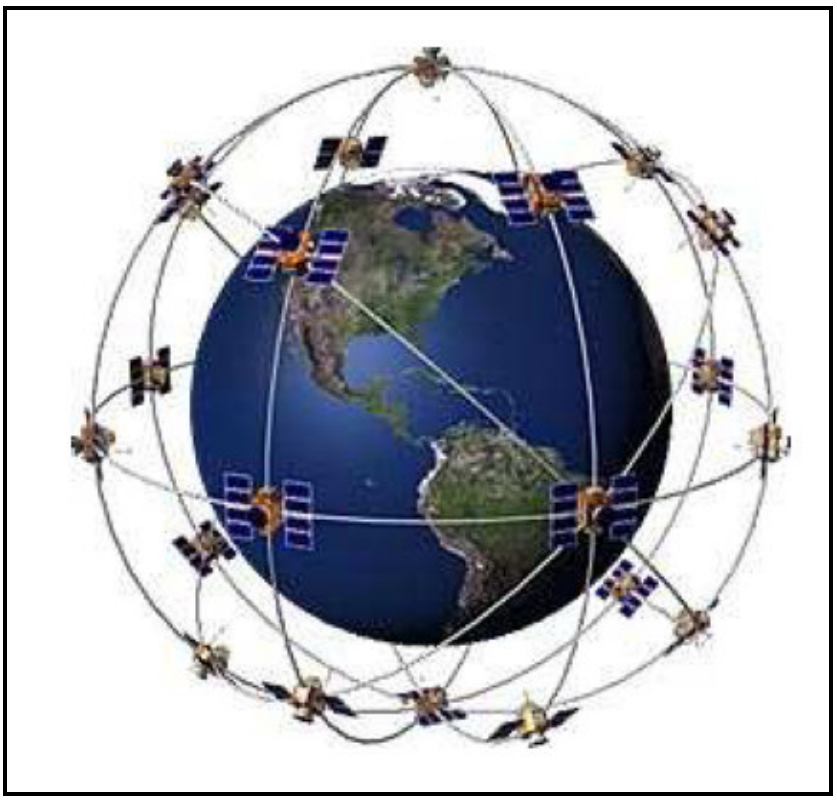

Figura 15 - Ilustração dos satélites em órbita
Fonte: site http://www.geocities.com/CapeCanaveral/Galaxy/5256/gps_introgarm.htm

Para computar uma posição através do sinal GPS podem ser utilizadas duas formas: dados de código (dados são computados a partir do código gerado pelo satélite e enviados ao usuário) e dados da portadora (utilizado o sinal de rádio que foi utilizado para transportar o sinal). Os receptores GPS de uso civil fazem o cálculo da posição atual através de dados de

código. A utilização de apenas um receptor é denominado posicionamento absoluto, esses receptores atuam na freqüência de L1 1.575,42 MHz e L2 1.227 MHz na banda UHF do espectro magnético (ROCHA, 2002).

O cálculo da posição se dá através dos sinais recebidos dos satélites, esses sinais são transmitidos em freqüências pré-definidas. As informações enviadas pelos satélites são: posição, horário de transmissão, os meios para o cálculo de distância até o satélite, parâmetros de correção das influências atmosféricas e o almanaque (dados de posição e *status* de todos os satélites da constelação GPS). Os receptores GPS necessitam visualizar os sinais de um mínimo de três satélites para calcularem continuamente a sua localização (IBIDEM).

O receptor ao receber os sinais dos satélites, calcula a distância para cada um destes, com base nos dados do almanaque, realizando triangulações como forma de estabelecer a sua posição. O receptor GPS terá quase sempre, de cinco a doze satélites em vista, e selecionará entre eles os melhores para o cálculo da posição. Para cálculos de posições tridimensionais são necessários um mínimo de quatro satélites, com três satélites é possível calcular apenas a posição bidimensional (2D). Após o cálculo da posição, os receptores GPS fornecem várias informações como: coordenadas de localização, velocidade, azimute, hora, entre outras (IBIDEM).

A Figura 16 demonstra o recebimento do sinal pelo receptor GPS. É possível reparar que existem quatro satélites transmitindo sinal, isso tornará possível uma localização tridimensional.

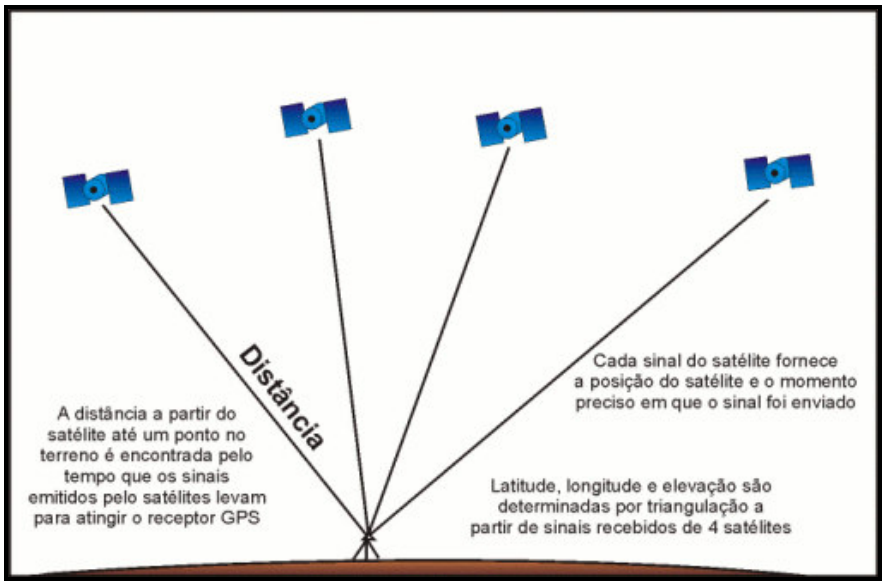
Distância
A distância a partir do satélite até um ponto no terreno é encontrada pelo tempo que os sinais emitidos pelo satélites levam para atingir o receptor GPS
Cada sinal do satélite fornece a posição do satélite e o momento preciso em que o sinal foi enviado
Latitude, longitude e elevação são determinadas por triangulação a partir de sinais recebidos de 4 satélites

Figura 16 - Cálculo da posição do receptor GPS
Fonte: site http://www.nara.org.br/servicos/ntp/gps/?searchterm=None

4.3 Receptores GPS

Pode ser considerado um receptor GPS qualquer equipamento capaz de receber as ondas eletromagnéticas, processar as informações recebidas e gerar uma saída. Atualmente existe uma grande variedade de receptores disponíveis no mercado, alguns prontos para usuários finais e outros para serem adicionados a outros produtos, no caso os módulos receptores de GPS (ROCHA, 2002).

Conforme Hoff (2005), os principais componentes de um receptor GPS são:

- **Antena**: tem a função de detectar as ondas eletromagnéticas emitidas pelo satélite, converter a energia da onda em corrente elétrica, amplificar o sinal e enviá-lo para a parte de eletrônica;
- **Seção de radiofreqüência**: os sinais são convertidos para uma freqüência mais baixa, chamada freqüência intermediária;
- **Canais**: A maioria dos receptores dispõe de seis a doze canais dedicados com capacidade para rastrear todos os satélites visíveis;
- **Microprocessador**: responsável por obter, processar e decodificar a mensagem de navegação, além de calcular a posição e outras informações;

- **Suprimentos de energia**: Todo receptor GPS necessita de uma fonte de energia.

Algumas características são reservadas apenas para receptores GPS de usuários finais, entre eles destaca-se:

- **Interface com o Usuário**: Trata-se da capacidade de interagir com o usuário através de entradas de teclado, displays, entre outros;
- **Memória**: para realizar diversos registros como: informações de rota e configurações.

4.4 *GPS Receiver Engine Board* ET-102/ER-102

O módulo GPS *Receiver Engine Board* ET-102/ER-102 desenvolvido pela *Globalsat Technology Corporation* localizada em Taiwan, será utilizado no desenvolvimento do projeto. É importante especificar a diferença entre um receptor GPS e um módulo receptor GPS, a grande diferença está no fato de o receptor GPS ser um produto destinado ao usuário final, enquanto que o módulo receptor GPS é destinado ao desenvolvimento de produtos no regimento OEM (*Original equipment manufacturer*). A Tabela 4 demonstra as principais características desse módulo (ET102/ER102).

Tabela 4 - Principais características do módulo receptor GPS ET-102/ER-102

Output Messages	NMEA0183 V2.2 *protocol, and supports command*: GGA,GSA,GSV,RMC,GLL,VTG
Hot Start	8 *sec.,average*
Warm start	38 *sec.,average*
Cold Start	45 *sec., average*
Acceleration Limit	< 4g
Altitude Limit	18,000m(60000 feet)Max
Re-acquisition	100ms
Velocity Limit	515 metros/segundo (1000 knots) Max
Jerk Limit	20 m/s **3
Horizontal Accuracy	15m 2d RMS without SA 5m 2d RMS WAAS enabled 1~ 5m DGPS corrected
Vertical Accuracy	40m 95%(SA off)

	5m WAAS enabled
Velocity Accuracy	0.1m/sec 95%(SA off)
Time Accuracy	1us synchronized to GPS time
Datum	WGS84
Dimension	71mm* 41mm* 9mm
Power consumption	60mA typical, trickle power mode 25mA
Storage Temperature	-40°C~85°C
Operation Temperature	-40°C~80°C
Humidity	Up to 95% non-Condensing
Frequency	*L1, 1575.42MHz*
C/A code	*1.023MHz chip rate*
Channels	12
Sensitivity	*-170dBW*
Temperature Operating	*-40 deg. C to +80 deg. C*

Fonte: User Manual GPS Receiver Engine Board ET-102/ ER-102

As características mais importantes desse módulo são: o fato da comunicação ser serial e a troca de mensagens utilizar o protocolo NMEA0183 com suporte as seguintes mensagens: GGA, GSA, GSV, RMC, GLL, VTG , esses padrões de mensagens serão estudados no tópico sobre protocolo NMEA.

Na Figura 17 pode ser visualizado a placa e os componentes eletrônicos que compõem o *GPS Receiver Engine Board* ET-102/ ER-102.

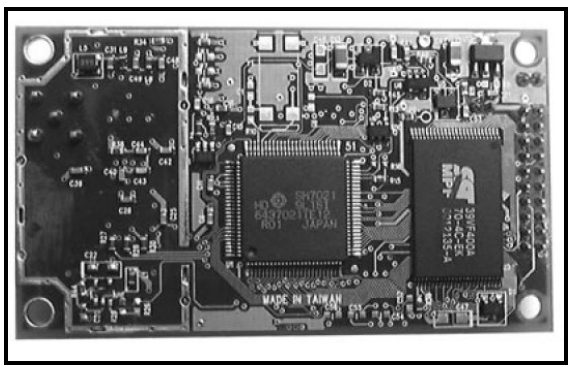

Figura 17 - GPS Receiver Engine Board ET-102/ER-102
Fonte: *User manual GPS Receiver Engine Board*: ET102/ER102

Esse módulo possui 20 pinos, tendo as seguintes funções exibidas na Tabela 5:

Tabela 5 – Distribuição de pinos para acesso ao módulo receptor GPS ET-102/ ER-102

Número do Pino	Nome	Descrição	Fluxo
6 / 8 / 9 / 17 / 20	Reservados	Reservados	
1	VANT	*Antenna DC Voltage*	Entrada
2	VDC	*3.8V~6.5V DC Power Input*	Entrada
3	VBAT	*Backup Battery*	Entrada
4	VDC	*(Shorted with pin 2)*	Entrada
5	PBRES	*Push Button Reset Input (Active Low)*	Entrada
10 / 13 / 16 / 18	GND	*Ground*	
11	TXA	*Serial Data Output A (GPS Data)*	Saída
12	RXA	*Serial Data Input A (Command)*	Entrada
19	TIMEMARK	*1PPS Time Mark Output*	Saída

Fonte: User Manual GPS *Receiver Engine Board* ET-102/ ER-102

4.5 *NMEA Output Command*

Nesta seção serão analisadas as saídas geradas pelo protocolo NMEA. Vale a ressalva de que as mensagens de saída habilitadas no módulo receptor GPS são: GGA, GSA, GSV, RMC, GLL, VTG. As mensagens de saída possuem alguns padrões, a mensagem ANSI de saída sempre iniciará com: "$GP" mais o identificador de protocolo. Os campos de dados são separados por ",", e o *checksum* é reconhecido pela antecedência do caráter "*" (NMEA, 2001).

O desenvolvimento de rotinas para capturar os valores presentes nas mensagens se torna viável pela forma com que os dados são transmitidos. Na maioria dos protocolos conhecidos, as informações e valores seguem posições fixas. O cabeçalho IP pode ser citado como exemplo, porém a estrutura das mensagens do protocolo NMEA se utiliza da separação dos dados por vírgula, bastando apenas a identificação do tipo de mensagem para que seja possível reconhecer os valores presentes. Quando um determinado parâmetro na mensagem não contiver valor, esse ficará em branco e será percebida na mensagem uma seqüência de duas vírgulas.

As mensagens possuem um *checksum* a ser calculado para aferir se as informações presentes estão corretas. Na Figura 18 é possível observar um exemplo de código em C que realiza o cálculo do *checksum*, essa função receberá como parâmetros a mensagem e o número de caracteres e retornará o cálculo do *checksum*. Dessa forma basta comparar o *checksum* recebido com o calculado para afirmar se a mensagem é valida ou não (IBIDEM).

```
unsigned char Calc_checksum(char* sz, int nCount)
{

  unsigned char cs;         // Checksum

  //Omit the $-character
  for (i=1; i<Count; i++)
  {
       cs = cs ^((unsigned char)sz[i]);
  }

  return cc;

}
```

Figura 18 - Cálculo do checksum na linguagem C
Fonte: *User manual GPS Receiver Engine Board*: ET102/ER102

Considerando que, é de extrema importância o conhecimento do protocolo de troca de mensagens entre o módulo receptor GPS e o processador ARM7, nos próximos tópicos serão abordados e estudados as mensagens de saída.

4.5.1 GGA-*Global Positioning System Fixed Data*

A mensagem GGA é uma das saídas geradas pelo receptor GPS, onde as principais informações extraídas dessa mensagem são: horário, posicionamento global e número de satélites. Abaixo é possível observar a estrutura da mensagem retornada pelo receptor GPS e um exemplo de retorno com valores. A mensagem inicia com a identificação e na seqüência os campos são separados por vírgulas.

$GPGGA,hhmmss.dd,xxmm.dddd,<N|S>,yyymm.dddd,<E|W>,v,ss,d.d,h.h,M,g.g,M,a
.a,xxxx*hh<CR><LF>

$GPGGA,161229.487,3723.2475,N,12158.3416,W,1,07,1.0,9.0,M,,,,0000*18

A Tabela 6 faz a identificação e detalhamento de todos os parâmetros presentes na estrutura da mensagem GGA. Na primeira coluna são identificados os parâmetros presentes na mensagem, na segunda coluna é apresentado um exemplo de valor possível para cada campo, a terceira coluna especifica o tipo de unidade e na última coluna é dada a descrição do campo e possíveis valores de retorno.

Tabela 6 - Formato GGA do protocolo NMEA

Nome	Exemplo	Unidade	Descrição
Message ID	$GPGGA		GGA *protocol header*
UTC Time	161229.487		hhmmss.sss
Latitude	3723.2475		ddmm.mmmm
N/S *Indicator*	N		N=north or S=south
Longitude	12158.3416		dddmm.mmmm
E/W *Indicator*	W		E=east or W=west
Position Fix Indicator	1		0 - *Fix not available or invalid* 1 - *GPS SPS Mode, fix valid* 2 - *Differential GPS, SPS Mode , fix valid* 4 -*GPS PPS Mode, fix valid*
Satellites Used	07		Range 0 to 12
HDOP	1.0		*Horizontal Dilution of Precision*
MSL Altitude	9.0	Metros	
Units	M	Metros	
Geoid Separation		Metros	
Units	M	Metros	
Age of Diff. Corr.		Segundos	Null fields when DGPS is not used
Diff. Ref. Station ID	0000		
Checksum	*18		
<CR><LF>			*End of message termination*

Fonte: User Manual GPS Receiver Engine Board ET-102/ ER-102

4.5.2 GLL-*Geographic Position-Latitude/Longitude*

A mensagem GLL é outra saída gerada pelo receptor GPS, as principais informações extraídas dessa mensagem são: posicionamento global, horário e *status*, o campo *status* é utilizado para indicar a validade das informações presentes na mensagem. Abaixo é possível

observar a estrutura da mensagem retornada pelo receptor GPS e um exemplo de retorno com valores. A mensagem inicia com a identificação do protocolo e na seqüência os campos são separados por vírgulas.

$GPGLL,xxmm.dddd,<N|S>, yyymm.dddd,<E|W>,hhmmss.dd,S,M*hh<CR><LF>

$GPGLL,3723.2475,N,12158.3416,W,161229.487,A*2C

A Tabela 7 faz a identificação e detalhamento de todos os parâmetros presentes na estrutura da mensagem GLL. Na primeira coluna são identificados os parâmetros presentes na mensagem, na segunda coluna é apresentado um exemplo de valor possível para cada campo, a terceira coluna especifica o tipo de unidade quando necessário e na última coluna é informada a descrição do campo e possíveis valores de retorno.

Tabela 7 - Formato GLL do protocolo NMEA

Nome	Exemplo	Unidade	Descrição
Message ID	$GPGLL		GLL *protocol header*
Latitude	3723.2475		ddmm.mmmm
N/S *Indicator*	N		N=north or S=south
Longitude	12158.3416		Dddmm.mmmm
E/W *Indicator*	W		E=east or W=west
UTC *Position*	161229.487		hhmmss.sss
Status	A		A=*data valid* or V=*data not valid*
Checksum	*2C		
<CR><LF>			*End of message termination*

Fonte: User Manual GPS Receiver Engine Board ET-102/ ER-102

4.5.3 GSA-*GNSS DOP and Active Satellites*

A mensagem GSA tem como principal finalidade informar se o modo de operação é automático ou forçado em 2D ou 3D, e a identificação dos satélites conectados nos canais do receptor. Abaixo é possível observar a estrutura da mensagem retornada pelo receptor GPS e um exemplo de retorno com valores. A mensagem inicia com a identificação do protocolo e na seqüência os campos são separados por vírgulas.

$GPGSA,a,b,xx,xx,xx,xx,xx,xx,xx,xx,xx,xx,xx,xx,p.p,h.h,v.v*hh<CR><LF>

$GPGSA,A,3,07,02,26,27,09,04,15,,,,,,1.8,1.0,1.5*33

A Tabela 8 faz a identificação e detalhamento de todos os parâmetros presentes na estrutura da mensagem GSA. Na primeira coluna são identificados os parâmetros presentes na mensagem, na segunda coluna é apresentado um exemplo de valor possível para o parâmetro e na terceira coluna é informada a descrição do campo e possíveis valores de retorno.

Tabela 8 - Formato GSA do protocolo NMEA

Nome	Exemplo	Descrição
Message ID	$GPGSA	GSA *protocol header*
Mode1	A	M= *Manual-forced to operate in 2D or 3D mode* A= *Automatic-allowed to automatically switch 2D/3D*
Mode2	3	1= *Fix Not Available* 2= 2D 3=3D
Satellite Used	07	*Sv on Channel 1*
Satellite Used	02	*Sv on Channel 2*
...		
Satellite Used		*Sv on Channel 12*
PDOP	1.8	*Position dilution of Precision*
HDOP		*Horizontal dilution of Precision*
VDOP		*Vertical dilution of Precision*
Checksum	*33	
<CR><LF>		*End of message termination*

Fonte: User Manual GPS *Receiver Engine Board* ET-102/ ER-102

4.5.4 GSV-*GNSS Satellites in View*

A mensagem GSV contém informações interessantes para o projeto como: número de satélites, *azimuth* (graus em relação ao satélite variando de 0 a 359). Abaixo é possível

observar a estrutura da mensagem retornada pelo receptor GPS e um exemplo de retorno com valores. A mensagem inicia com a identificação do protocolo e na seqüência os campos são separados por vírgulas.

$GPGSV,n,m,ss,xx,ee,aaa,cn,…… ,xx,ee,aaa,cn*hh<CR><LF>

$GPGSV,2,1,07,07,79,048,42,02,51,062,43,26,36,256,42,27,27,138,42*71

A Tabela 9 faz a identificação e detalhamento de todos os parâmetros presentes na estrutura da mensagem GSV. Na primeira coluna são identificados os parâmetros presentes na mensagem, na segunda coluna é apresentado um exemplo de valor possível para cada campo, a terceira coluna especifica o tipo de unidade quando necessário e na última coluna é informada a descrição do campo e possíveis valores de retorno.

Tabela 9 - Formato GSV do protocolo NMEA

Nome	Exemplo	Unidade	Descrição
Message ID	$GPGSV		GSV *protocol header*
Number of Messages	2		Range 1 to 3
Message Number	1		Range 1 to 3
Satellites in View	07		
Satellite ID	07		*Channel* 1(Range 1 to 32)
Elevation	79	*degrees*	*Channel* 1(Maximum90)
Azimuth	048	degrees	*Channel* 1(*True*, Range 0 to 359)
SNR(C/No)	42	dBHz	Range 0 to 99,null when not tracking
Satellite ID	27		*Channel* 4 (Range 1 to 32)
Elevation	27	degrees	*Channel* 4(Maximum90)
Azimuth	138	degrees	*Channel* 4(*True*, Range 0 to 359)
SNR(C/No)	42	dBHz	Range 0 to 99,null when not tracking
Checksum	*71		
<CR><LF>			End of message termination

Fonte: User Manual GPS Receiver Engine Board ET-102/ ER-102

4.5.5 RMC-*Recommended Minimum Specific GNSS Data*

A mensagem RMC contém vários parâmetros importantes como: horário, posicionamento global, data e velocidade. Abaixo é possível observar a estrutura da mensagem retornada pelo receptor GPS e um exemplo de retorno com valores. A mensagem inicia com a identificação do protocolo e na seqüência os campos são separados por vírgulas.

$GPRMC,hhmmss.dd,S,xxmm.dddd,<N|S>,yyymm.dddd,<E|W>,s.s,h.h,ddmmyy,d.d,<E|W>,M*hh<CR><LF>

$GPRMC,161229.487,A,3723.2475,N,12158.3416,W,0.13,309.62,120598,,*10

A Tabela 10 faz a identificação e detalhamento de todos os parâmetros presentes na estrutura da mensagem RMC. Na primeira coluna são identificados os parâmetros presentes na mensagem, na segunda coluna é apresentado um exemplo de valor possível para cada campo, a terceira coluna especifica o tipo de unidade quando necessário e na última coluna é informada a descrição do campo e possíveis valores de retorno.

Tabela 10 - Formato RMC do protocolo NMEA

Nome	Exemplo	Unidade	Descrição
Message ID	$GPRMC		RMC *protocol header*
UTC Time	161229.487		hhmmss.sss
Status	A		A=*data valid* or V=*data not valid*
Latitude	3723.2475		ddmm.mmmm
N/S *Indicator*	N		*N= North or S= South*
Longitude	12158.3416		dddmm.mmmm
E/W *Indicator*	W		E=*east* or W=*west*
Speed Over Ground	0.13	*Knots*	
Course Over Ground	309.62	Knots	True
Date	120598		ddmmyy
Magnetic Variation		*Degrees*	*E= East or W= West*
Checksum		*10	
<CR><LF>			*End of message termination*

Fonte: User Manual GPS Receiver Engine Board ET-102/ ER-102

4.5.6 VTG-*Course Over Ground and Ground Speed*

A principal informação contida na mensagem VTG é a velocidade. Abaixo é possível observar a estrutura da mensagem retornada pelo receptor GPS e um exemplo de retorno com valores. A mensagem inicia com a identificação do protocolo e na seqüência os campos são separados por vírgulas.

$GPVTG,h.h,T,m.m,M,s.s,N,s.s,K,M*hh<CR><LF>

$GPVTG,309.62,T,,M,0.13,N,0.2,K*6E

A Tabela 11 faz a identificação e detalhamento de todos os parâmetros presentes na estrutura da mensagem VTG. Na primeira coluna são identificados os parâmetros presentes na mensagem, na segunda coluna é apresentado um exemplo de valor possível para cada campo, a terceira coluna especifica o tipo de unidade quando necessário e na última coluna é informada a descrição do campo e possíveis valores de retorno.

Tabela 11 - Formato VTG do protocolo NMEA

Nome	Exemplo	Unidade	Descrição
Message ID	$ GPVTG		VTG *protocol header*
Course	309.62	degrees	*Measured heading*
Reference	T		*True*
Course		degrees	*Measured heading*
Reference	M		*Magnetic*
Speed	0.13	Knots	*Measured horizontal speed*
Units	N		*Knots*
Speed	0.2	Km/hr	*Measured horizontal speed*
Units	K		*Kilometers per hour*
Checksum	*6E		
<CR><LF>			*End of message termination*

Fonte: User Manual GPS Receiver Engine Board ET-102/ ER-102

4.6 *NMEA Imput Command*

Os comandos de entrada são utilizados para determinar as configurações e formas de funcionamento do módulo receptor GPS, esses comandos utilizam o protocolo NMEA. O padrão dos comandos está definido da seguinte forma: $PSRF indica o início de um comando, os parâmetros a serem enviados ao módulo receptor GPS são separados por vírgulas, e o fim da mensagem ANSI é determinada pelo "*" mais o *checksum*.

Conforme o manual ET-102/ ER-102, os comandos mais importantes a serem enviados para o módulo receptor GPS são:

- **Configurar a porta serial**: Para utilizar a porta serial é necessário definir os parâmetros de configuração. O seguinte padrão de mensagem é utilizado: $PSRF100, <protocol>, <baud>, <DataBits>, <StopBits>, <Parity>*CKSUM<CR><LF>;
- **Inicia a navegação:** comando utilizado para iniciar o módulo para *warm start,* os seguintes parâmetros devem ser passados: $PSRF101, <X>, <Y>, <Z>, <ClkOffset>, <TimeOfWeek>, <WeekNo>, <chnlCount>, <ResetCfg*CKSUM<CR><LF>;
- **Definir mensagem NMEA de saída:** Esse comando é utilizado para definir o tipo de mensagem de saída. Os seguintes padrões podem ser definidos: GGA, GSA, GSV, RMC, GLL, VTG. A seguinte mensagem deve ser enviada para habilitar esses modos: $PSRF103, <msg>, <mode>, <rate>, <cksumEnable>*CKSUM<CR><LF>;
- **Ativar modo de depuração:** Essa configuração permite depurar as mensagens enviadas, e caso ocorra alguma inconformidade é retornado o erro. Para ativar esse módulo a seguinte mensagem deve ser enviada: $PSRF105,<debug>*CKSUM<CR><LF>.

O receptor GPS é um item fundamental no projeto proposto, pois dele virão todas as informações necessárias para a classificação do condutor. Fica inevitável o estudo do Sistema de Posicionamento Global (GPS), protocolo NMEA, e as especificações técnicas referentes ao receptor GPS utilizado no projeto, pois esses estudos darão o conhecimento necessário para desenvolver o projeto.

5 SISTEMAS INTELIGENTES

Para tratar a informação de forma inteligente é necessário a implementação de um sistema inteligente, que trate a informação recebida do receptor GPS e forneça saídas compreensíveis ao ser humano. Nos próximos parágrafos serão apresentados alguns algoritmos que poderiam ser utilizados e sua relevância em relação aos resultados.

Uma solução possível seria o uso de um algoritmo que utilizasse a lógica booleana, porém essa lógica são se aplica muito bem no processo de classificação do condutor pelo fato de ser abrupta e não tratar informações imprecisas, ou seja, uma informação só pode ser verdadeira ou falsa (NETO, et al., 2006). Um exemplo da lógica booleana pode ser apresentado da seguinte forma: Se a temperatura da água for igual ou maior que 70 graus Celsius então a água está quente, caso contrário ela está fria, seria correto afirmar que a água a 69 graus Celsius estaria fria?

Sistemas especialistas talvez pudessem resolver o problema proposto neste projeto, porém o sistema especialista tem como peculiaridade simular o conhecimento humano parecendo de certa forma inteligente. Para que seja implementado um sistema especialista é necessário traduzir o conhecimento de um especialista, fato que gera uma vasta base de conhecimento sobre um domínio. Sistemas especialistas geralmente são utilizados como sistemas de auxílio para profissionais de uma determinada área. O grande problema da utilização dos sistemas especialistas está relacionado ao tratamento de dados imprecisos (IBIDEM).

A lógica *fuzzy* se mostrou a mais apropriada para o desenvolvimento do projeto por possibilitar o trabalho com informações imprecisas, um exemplo poderia ser a utilização de várias determinações para uma única variável, considerando que a variável seja velocidade, poderíamos assumir as seguintes valores: muito baixa, baixa, média, alta, muito alta. Nos próximos tópicos será realizado um estudo mais aprofundado na lógica *fuzzy*.

5.1 Lógica *Fuzzy*

Lotfi Zadeh, professor de engenharia elétrica e ciências da computação desenvolveu em 1965 a teoria da lógica de *fuzzy*. Zadeh observou que muitas das regras utilizadas pelas pessoas para fazer inferência não podiam ser explicadas por elas. Um exemplo disso seria dizer: "Aquele homem deve ter 40 anos", não estamos preparados para explicar as regras utilizadas para chegar nessa determinação, baseado nessas observações Zadeh criou a lógica *fuzzy*, inicialmente criticada, porém acabou sendo utilizada por engenheiros e cientistas da computação (COSTA, et al, 2005).

Tratar informações subjetivas e imprecisas ou até mesmo ambíguas é algo natural para a mente humana, porém computadores foram construídos e preparados para trabalharem com informações precisas e bem estruturadas, não podendo reconhecer informações do tipo: está um pouco quente hoje, está quase pronto em sua forma inicial. Tendo em vista esses conceitos a lógica *fuzzy* surgiu para que essas informações pudessem ser tratadas (JÚNIOR; NOVAKOWSKI, 2005).

Na Figura 19 é possível observar claramente a diferença entre a lógica clássica e a lógica *fuzzy*, a lógica clássica não abre espaço para valores intermediários enquanto na lógica *fuzzy* é possível analisar os valores intermediários. Um exemplo clássico seria: Na lógica clássica o dia pode estar frio ou quente enquanto que na lógica *fuzzy* pode-se ter várias determinações como: frio, muito frio, pouco frio, morno, quente, muito quente, pouco quente entre outras.

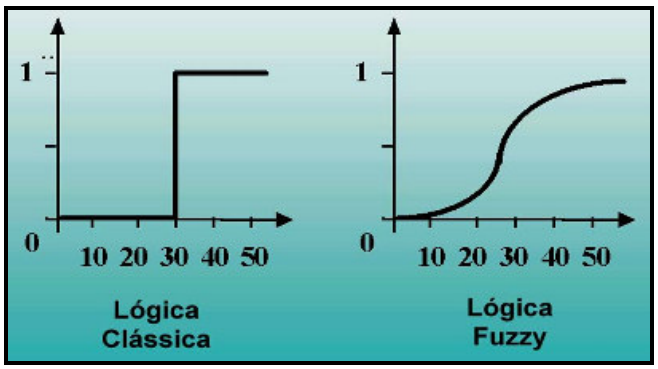

Figura 19 - Diferença entre lógica clássica e lógica *fuzzy*
Fonte: apresentação Marcílio Souto

5.1.1 Teoria dos conjuntos *fuzzy*

Os conjuntos *fuzzy* foram desenvolvidos para determinar o quanto um elemento pertence ou não a um determinado conjunto, com o uso de graus de pertinência (μ) que são valores no intervalo [0;1]. Assim é possível determinar os elementos que pertencem a um grupo e seu respectivo grau de pertinência (JÚNIOR; NOVAKOWSKI, 2005).

A representação dos conjuntos *fuzzy* depende da natureza e das dimensões do universo a ser representado. Quando for necessário representar conjuntos que contenham um universo pequeno, a melhor solução é a analítica, que tem o objetivo de apresentar todos os elementos do universo separados por vírgula, sendo que cada elemento é composto pelo grau de pertinência e o valor do termo. Quando o universo a ser representado contiver um grande número de elementos, ele deve ser representado pelo gráfico de sua função de pertinência, chamado de diagrama de Hassi-Euler (H-E) (REZENDE, 2003).

A Figura 20 apresenta um exemplo de determinação de conjuntos *fuzzy*. É possível perceber a representação do universo dos elementos no item 1 da figura, sendo esse universo composto por idades. No item 2 da figura são criados três conjuntos, sendo eles jovens, adultos e velhos. Já no item 3 da figura são representados os graus de pertinência aos grupos em relação aos elementos do universo. É possível reparar que aos trinta anos um indivíduo tem um grau de pertinência de 0,5 em relação ao conjunto jovem, 0,4 em relação ao adulto e 0,1 em relação ao velho.

1) U = {5, 10, 20, 30, 40, 50, 60, 70}

2) Conjuntos difusos: jovens, adultos, velhos

3)

Idades	Jovens	Adultos	Velhos	(μ)
5	1,0	0,0	0,0	
10	1,0	0,0	0,0	
20	0,8	0,2	0,0	
30	0,5	0,4	0,1	
40	0,2	0,4	0,4	
50	0,1	0,3	0,6	
60	0,0	0,2	0,8	
70	0,0	0,0	1,0	

Figura 20 - Exemplo do uso de conjuntos *fuzzy*
Fonte: Material de aula Rejane Frozza

5.1.2 Variável lingüística

Variável lingüística é a entidade utilizada para representar de forma imprecisa um conceito ou uma variável de um determinado problema. Essa variável aceita somente valores lingüísticos, tais como: frio, pouco frio, muito frio, grande, muito grande (REZENDE, 2003). Analisando-se a Figura 20 pode-se reconhecer a idade como sendo a variável lingüística e os seus valores representados pelos conjuntos difusos: jovem, adulto e velho, esses valores contrastam com valores numéricos.

O processo de fuzzificação é responsável por atribuir a cada conjunto difuso os respectivos valores de pertinência (JÚNIOR; NOVAKOWSKI, 2005). Na Figura 21 é possível observar um exemplo da atribuição dos valores de pertinência aos valores referentes à variável lingüística velocidade. Observando-se a Figura 21, pode-se notar que a velocidade de 30 km/h tem grau de pertinência 0,5 tanto para o valor devagar quanto para médio, em contrapartida pode-se reparar que a velocidade de 60 km/h tem grau máximo de pertinência em relação ao valor médio e nulo em relação aos valores devagar e rápido.

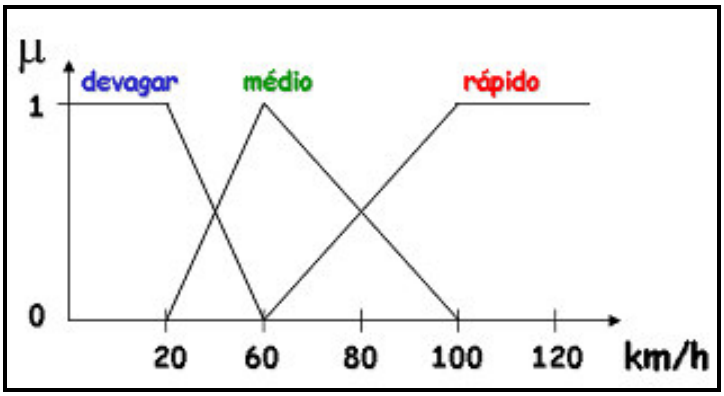

Figura 21 - Exemplo de fuzzificação
Fonte: material de aula Rejane Frozza - UNISC

5.1.3 Sistema de inferência *fuzzy*

O sistema de inferência *fuzzy* ou controlador *fuzzy* é composto de três fases, sendo elas: fuzzificação, inferência e defuzzificação. Na Figura 22 é possível visualizar o sistema de inferência *fuzzy* e a forma de interligação entre as fases, sendo que cada fase tem como princípio uma entrada de dados, processamento sobre esses dados e uma saída.

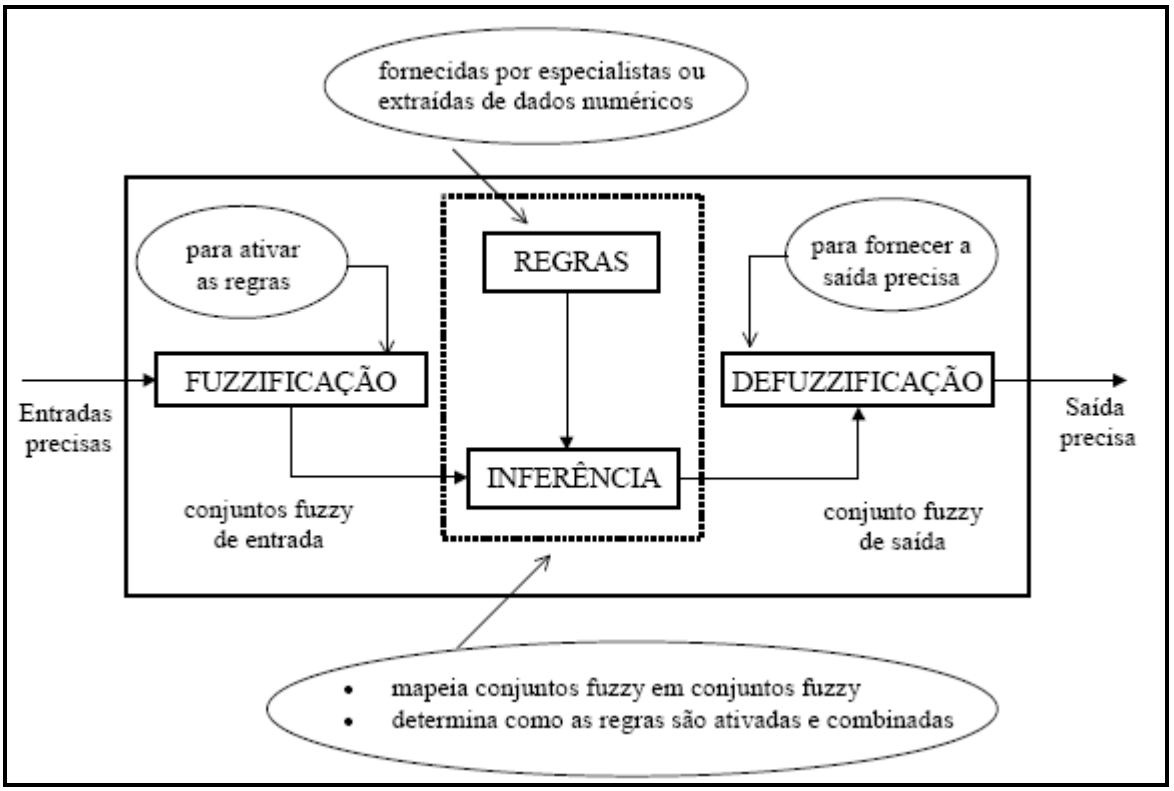

Figura 22 - Diagrama de blocos de um controlador *fuzzy* típico
Fonte: http://www.ica.ele.puc-rio.br/cursos/download/LN-Sistemas%20Fuzzy.pdf

O primeiro passo realizado no sistema de inferência *fuzzy* é a fuzzificação, esse processo deve ser realizado para cada valor de entrada e consiste no mapeamento dos dados para conjuntos *fuzzy* relevantes. Os valores de entrada são valores precisos provenientes geralmente de medições ou observações. O processo de fuzzificação também é responsável pela ativação das regras relevantes para uma determinada situação (TANSCHEIT, 2008).

Em um controlador *fuzzy* as regras têm um papel relevante, pois o bom desempenho do sistema está vinculado à produção de regras consistentes. A base do conhecimento pode ser fornecida por especialistas através de termos lingüísticos. Esse processo pode ser complicado, mas, independente do conhecimento do especialista, uma alternativa para formação da base de regras é através da mineração de dados numéricos Esse método tem um bom retorno em problemas de classificação (IBIDEM).

A construção de uma base de conhecimento pode dar-se através de regras de produção *fuzzy*, sendo essa a maneira mais comum. Uma regra de produção é formada por duas partes principais: antecedente, conseqüente. A estrutura de uma regra de produção é definida da seguinte forma: “Se <antecedente> então <conseqüente>”, na Figura 23 é possível visualizar

um exemplo de regra de produção, a cláusula entre o se e o então é denominada antecedente e a sentença após do então é denominada conseqüente (REZENDE, 2003).

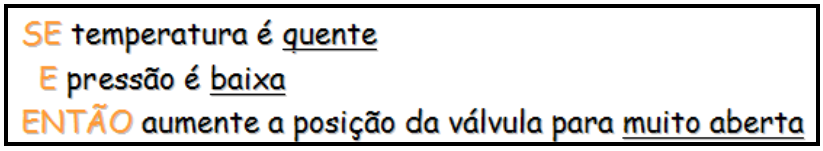

Figura 23 - Exemplo de regra de produção *fuzzy*
Fonte: material de aula Rejane Frozza - UNISC

A fase da inferência é responsável pela operação sobre os conjuntos propriamente dita. Nela são determinadas as regras que serão ativadas e combinadas e como resultado desse processo são produzidos conjuntos *fuzzy* de saída. (TANSCHEIT, 2008).

O sistema de inferência *fuzzy* na maioria das vezes deve disponibilizar dados precisos como saída. Isso se deve pelo fato da maioria das aplicações necessitarem dessa informação. A etapa de defuzzificação é responsável por interpretar as informações geradas pela fase da inferência. Existem vários métodos de defuzzificação, entre eles: (IBIDEM).

- **Método do máximo:** o valor de saída é o de pertinência mais alto;
- **Método do centróide:** centro da área ou centro da gravidade;
- **Método da média ponderada:** determina-se o centróide de cada função envolvida, depois se faz a relação entre a pertinência da função e a pertinência máxima. Esse método é restrito a funções de pertinência simétrica;
- **Método da média dos máximos:** Similar ao método dos máximos, aplicável onde existe um grupo de elementos com pertinência máxima;
- **Método do centro da soma:** Similar ao método da média ponderada. Os pesos são as áreas das respectivas funções de pertinência;
- **Método da maior área:** Utiliza-se o método centróide para calcular o centro da maior área.

Para realizar a classificação do condutor, utilizou-se a lógica *fuzzy,* após o seu estudo, a mesma se mostrou eficiente para a solução. A lógica *fuzzy* será implementada no processador ARM7, estudado no primeiro capítulo deste trabalho.

6 ANÁLISE COMPORTAMENTAL DOS MOTORISTAS

Para o desenvolvimento desse projeto é necessário que haja um estudo do comportamento dos motoristas no trânsito, o reconhecimento de padrões de atitudes e pensamentos dos condutores. Todas elas são importantes para a definição das regras a serem utilizadas no sistema proposto. Existem vários trabalhos realizados na área da psicologia do trânsito que demonstram as varias relações entre condutores e comportamentos. Somente analisando os estudos já desenvolvidos na área da psicologia do transito será possível desenvolver um projeto que realmente atenda o objetivo principal deste projeto.

Num passado recente o maior investimento na segurança ocorreu através dos profissionais de engenharia e ergonomia, profissionais que tornaram o veículo de transporte ou de passeio cada vez mais seguro, com exemplos dos esforços empregados, pode-se citar o cinto de segurança, barreiras de colisão nas rodovias, veículos que se deformam em colisões de forma a proteger os passageiros, *airbag*, barras laterais, sensores de estacionamento, lombadas eletrônicas, entre outros. Todos esses procedimentos reduziram o número de acidentes, feridos e mortos nas cidades e nas rodovias, porém essas medidas trataram a menor parte dos motivos de acidentes, o que corresponde a menos de 10 %, podendo esse percentual variar de pesquisa para pesquisa realizada (HOFFMAN, 2005).

Atualmente existe um interesse muito grande na área da psicologia do trânsito, uma vez que 90% ou mais dos acidentes são causados por erros ou infrações de transito, é visível que a intenção dos profissionais da psicologia é criar padrões referentes à forma de conduzir o veículo, tarefa nada fácil tendo em vista o enorme número de fatores que podem influenciar um motorista. Podemos citar como fatores o trânsito, ambiente, bebidas alcoólicas, atitudes de outros motoristas, pressa, distração, entre outros (IBIDEM).

Conforme Monteiro e Günter (2006) erros e violações de trânsito são intrínsecos no ambiente de trânsito e fazem parte do dia-a-dia de cada motorista, nenhum ou poucos são os motoristas que não cometem infrações e erros no trânsito. Em função da rapidez em que as decisões devem ser tomadas para evitar conflitos e acidentes, o trânsito se torna um local estressante e propício para agravos no trânsito, tendo em vista o tamanho da frota de veículos existente e sabendo de que raros são os motoristas 100% corretos.

Existem varias razões para que sejam cometidos erros e violações nas leis de trânsito, e os motoristas as cometem em função de uma ou mais delas: ambiente físico permite que o faça sem danificar o veiculo e a si mesmo, a fiscalização do cumprimento das normas não está sendo feito de forma adequada, o ambiente social do trânsito permite ou até incentiva tal comportamento e características pessoais (MONTEIRO; GÜNTER, 2006).

A elevada mortalidade em acidentes de trânsito é um problema de saúde pública em quase todos os países, sendo que o maior número de vítimas fatais encontra-se entre jovens do sexo masculino e as principais causas são apresentadas em uma pesquisa realizada no Canadá, como sendo: a falta de experiência, o limitado poder de decisão, excessiva velocidade e infrações. A tendência a acidentes de trânsito pode ser prognosticada com base em infrações anteriormente cometidas, e um bom indicador é o excesso de velocidade (MARIN-LEÓN; VIZZOTO, 2003). Percebendo que existe uma relação clara entre infrações e acidentes de trânsito, é possível utilizar esse conceito na classificação dos condutores.

Todo e qualquer excesso de velocidade é perigoso. Em estudos realizados é necessário um espaço de 26 metros para evitar um atropelamento ou colisão em uma velocidade de 50 Km/h, enquanto que são necessários 35 metros se a velocidade for de 60 Km/h. Com apenas uma aumento de 20% na velocidade a distância para evitar a colisão aumenta em 40%. Para minimizar os danos realizados pelo excesso de velocidade, o Brasil conta com a fiscalização eletrônica conhecida popularmente como lombadas e pardais. Os resultados apresentados em locais que adotaram a fiscalização eletrônica no Brasil demonstram uma redução de 30% dos acidentes e uma redução de 60% na mortalidade de trânsito. Em todos os locais em que houve a instalação de equipamentos de fiscalização eletrônica os resultados foram satisfatórios (THIELEN, et al., 2007).

Estima-se que 30% dos acidentes com vitimas fatais ou graves é em decorrência do excesso de velocidade. Pode-se argumentar que a alta velocidade, por si só, não acarreta o risco de velocidade, porém é difícil sustentar que as conseqüências sejam iguais em alta e baixa velocidade, assim como o tempo para a tomada de decisão é menor (HOFFMANN, 2005).

Em uma pesquisa realizada por Marin-León e Vizzoto (2003) com 1642 condutores de uma universidade pública da cidade de Campinas no estado de São Paulo, criaram diversas

relações entre ações no trânsito e o número de acidentados através da estatística gerada na pesquisa. Os entrevistados tinham entre 18 e 25 anos e cursavam o ensino superior e a proporção de homens para mulheres era de 57,4% para 42,6% e teriam que responder 50 perguntas. Os principais resultados obtidos foram em relação a antecedentes de acidentes de trânsito:

- 12,3% dos que se referiram a não dirigir em alta velocidade;
- 34,5% dos que cometiam alta velocidade;
- 50% dos entrevistados que disseram dirigir em alta velocidade por economia de tempo;
- 41,6% são contra a nova legislação de trânsito, que por sinal é mais rígida.
- 43% dizem que as causa dos acidentes são fenômenos externos como: excesso de veículos, tipo de via, pedestres, entre outros;

Outra informação interessante apresentada pela pesquisa foi o índice de reincidência em acidentes de trânsito, dos 491 estudantes com antecedentes de trânsito 65,2% informaram ter se envolvido em apenas um acidente de transito, 20,8% se envolveram em dois acidentes e 14% em três ou mais (MARIN-LEÓN; VIZZOTO, 2003). Pode-se notar que esses números são realmente chocantes, pois em se tratando de uma pesquisa com 1642 entrevistados que tinham a carteira nacional de habilitação há no máximo sete anos, 491 já haviam se envolvido em acidentes, o que representa um percentual de quase 30%.

Existem várias abordagens para mudança do comportamento do motorista, sendo elas: planejar o leiaute da pista para que o comportamento divergente não seja fisicamente viável, desenvolver punições severas que condenem atos sociais divergentes, aumentar a consciência do condutor sobre os riscos, remover da circulação motoristas reincidentes. A escolha de uma das opções vai depender da cultura de cada local. O aumento ou a diminuição do nível de aplicação da lei em níveis normais não faz efeito sobre o comportamento do condutor, porém quando essa aplicação ocorre de forma automática ela tende a alterar o comportamento do motorista (HOFFMANN, 2005).

Após a análise de vários trabalhos relacionados na área da psicologia do trânsito é possível criar diversos conceitos que serão ligados diretamente a esse projeto, nos próximos

parágrafos serão mencionados os pontos de conclusão entre os estudos da psicologia do trânsito e o presente projeto.

O primeiro ponto refere que o devido projeto direciona-se pelo caminho correto, uma vez que o seu foco é sobre o comportamento do motorista, que é responsável por mais de 90% dos acidentes de trânsito através de erros e infrações.

O segundo item é sobre fiscalização eletrônica automática, que avalia os seus benefícios em relação à redução dos acidentes e mortes fatais. Sobre esse item é possível deduzir que, se em uma rodovia existem pontos de controle de velocidade, e esses pontos na maioria das vezes são do conhecimento de quem transita nela com freqüência e assim mesmo ela surte efeito, então podemos supor que um monitoramento permanente tenha maiores efeitos. Imaginamos o seguinte: Todos os carros tenham o presente projeto acoplado de fábrica, onde, ao invés de se ter pardais nas rodovias, tem-se pontos de coleta, que ao passar por um determinado ponto o carro informasse automaticamente a média de velocidade dos últimos dez minutos. Nesse caso o receptor saberia qual a média máxima para o trecho nos últimos dez minutos e se o veículo acusasse uma superior a essa, seria automaticamente penalizado. Esse procedimento faria com que o motorista se mantivesse dirigindo de forma harmônica em relação as sinalizações da rodovia.

Nas pesquisas apresentadas foi possível perceber que um grande número de acidentes ocorre em função de distração do motorista. Essa distração pode ser fruto de muitos motivos como: família, negócios, festas ou até mesmo esgotamento físico. Para solucionar esse item seria interessante criar um mecanismo que alertasse o motorista através de sinais sonoros de forma aleatória, sem que o motorista pudesse prever ou sincronizar mentalmente os eventos de som. Não existem estudos que comprovem a eficácia desse método, porém ele iria tirar muitas pessoas do estado de hipnose.

Em todas as pesquisas estudadas e analisadas, ficou provado que os motoristas que cometem infrações de trânsito são mais propensos a se envolverem em acidentes de trânsito, no presente projeto é possível analisar diversas infrações como: arrancadas bruscas, freadas bruscas, excessos de velocidade, entre outros. E a partir dessas informações emitir saídas que estimulem o bom comportamento. Os estudos na área da psicologia do trânsito são necessárias para fomentar as regras utilizadas na lógica *fuzzy*.

7 DESENVOLVIMENTO DO PROJETO

Neste capítulo será descritos os detalhes sobre o desenvolvimento do presente projeto.

7.1 Estrutura do projeto

A estrutura do sistema que visa desenvolver a aplicação de classificação de condutores é baseada em hardware e software. Os principais equipamentos presentes no projeto são: receptor GPS e o processador ARM7 acoplado na placa de desenvolvimento da Keil. Os principais itens de software presentes no projeto são: Comunicação com receptor GPS, classificação do condutor utilizando a lógica *fuzzy*.

Na Figura 24 é possível obter uma visão macro da relação entre os itens de hardware e software. Analisando a imagem pode-se observar a ligação entre o receptor GPS e os satélites. O receptor GPS é capaz de proporcionar várias informações como: posição global, velocidade, aceleração, *azimuth* entre outras em frações de tempo equivalentes a um segundo.

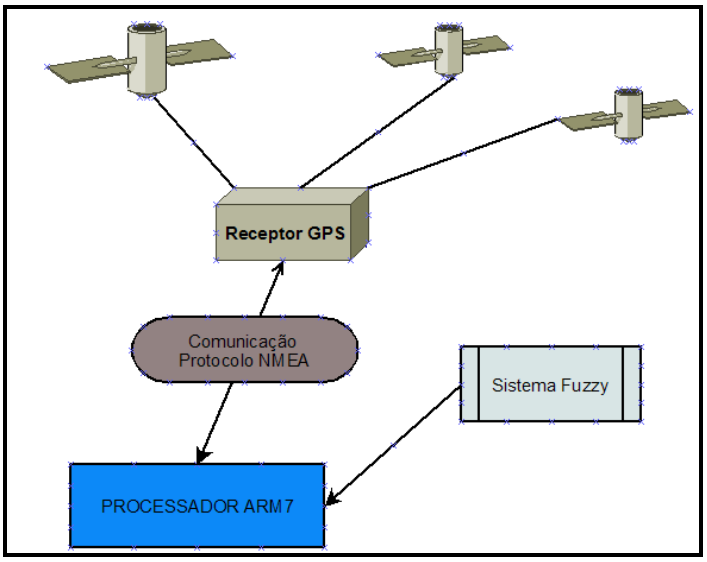

Figura 24 - Estrutura do projeto de classificação de condutores
Fonte: do autor

Para recuperar as informações do receptor GPS é utilizado o protocolo de comunicação NMEA, lembrando que esse protocolo além de possibilitar a recuperação da

informação também permite através da sua interface a configuração do módulo receptor GPS, em nível físico essa comunicação entre a placa de desenvolvimento e o receptor GPS se dará através de comunicação serial RS232.

A informação sem tratamento de nada adiantará para o desenvolvimento do projeto. Para que ela realmente tenha valor é necessário minerar e organizar esses dados de forma a obter os resultados esperados. Para realizar essa tarefa será utilizado a lógica *fuzzy*, que será implementada no processador ARM7. Vale a ressalva de que a lógica de fuzzy é especializada em operar com valores imprecisos

O conjunto de regras para a classificação dos condutores está presente no controlador *fuzzy* a ser implementado no projeto. A determinação de regras eficientes e consistentes depende de uma triangulação de fatores sendo eles:

1)Estatísticas, artigos e documentários desenvolvidos por especialistas da área. Nesta etapa o projeto conta com o apoio do professor do curso de psicologia da UNISC, Marcus Vinicius Castro Witczak.

2)Base de dados obtidas da leitura do receptor GPS para determinar os limites desejáveis. Um exemplo possível de se apresentar seria: supunha que haja a necessidade de classificar a arrancada de um veículo em quatro níveis: lenta, normal, rápida e agressiva. Com base nesses dados a pergunta é: Quais são os valores limites para definir cada um dos níveis?

3)CTB (Código de Trânsito Brasileiro). A partir deste documento oficial que rege as leis a serem praticadas no trânsito será possível a extração de informações importantes para a composição das regras do sistema *fuzzy*.

7.2 Configuração da velocidade de processamento

Uma das principais tarefas a ser executada no início de um programa é a configuração do *clock* de processamento. Na Figura 25 é possível observar o código que realiza essa tarefa

no programa deste projeto, que configura o *clock* de processamento para 60 MHz e o *clock* do barramento VPB, onde estão conectados os periféricos para 30 MHz (ScienceProg, 2008).

```
void init_PLL_VPB(void){

    PLLCFG  = 0x00000024;
    PLLCON  = 0x00000001;

    PLLFEED = 0x000000AA;
    PLLFEED = 0x00000055;

    while(!(PLLSTAT & 0x00000400));

    PLLCON = 0x00000003;

    PLLFEED = 0x000000AA;
    PLLFEED = 0x00000055;

    VPBDIV  = 0x00000002;

}
```

Figura 25 - Configuração do *clock* de processamento
Fonte: do autor

Para realizar a configuração é necessário conhecer o oscilador presente na placa de desenvolvimento, sendo que o kit da Keil é equipado com um cristal de 12MHz. Para que o processador utilize o máximo de sua potência se deve configurar o multiplicador por 5, esse valor surge da divisão do *clock* do processador pelo valor do cristal (60/12=5). Para se obter o divisor é necessário realizar: (clock * 2 * D), onde o resultado deve ficar entre 156MHz e 320Mhz. Os possíveis valores para o divisor são: 1,2,4,8. Para esse caso, o único valor que se enquadra no cálculo corretamente é o 2, onde (60 * 2 * 2 = 240), e 240 está no intervalo autorizado (IBIDEM).

Após encontrados os respectivos valores de 5 para o multiplicador e 2 para o divisor, é o momento de se configurar o registrador do *clock* do processador. Na Figura 26 pode-se visualizar uma tabela que ajudará nessa tarefa. Os bits 5 e 6 são utilizados para a configuração do divisor e os bits 0 ao 4 são utilizados para configurar o multiplicador. Sendo assim, basta unir os nossos dois valores em binário: 01 00100 onde obtêm-se o valor 36 em decimal, sendo o seu correspondente em hexadecimal 24.

PSEL Bits (PLLCFG bits [6:5])	Value of P
00	1
01	2
10	4
11	8

MSEL Bits (PLLCFG bits [4:0])	Value of M
00000	1
00001	2
00010	3
00011	4
...	...
11110	31
11111	32

Figura 26 - Tabela de configuração do registrador PLLCFG.
Fonte: do autor

O registrador PPLCON é utilizado para habilitar o módulo PLL, sendo esse registrador configurado através de 2 bits, o primeiro habilita o PLL e o segundo estabelece a conexão do PLL. Analisando a Figura 25, percebe-se que inicialmente esse é habilitado através do valor 0x01 que corresponde em binário a 01 e no final da rotina recebe o valor 0x03 que em binário corresponde a 11.

O registrador PLLFEED é utilizado para recarregar os valores dos registradores anteriormente configurados. Para realizar essa tarefa, basta carregá-lo com valor 0xAA e posteriormente com 0x55 (SOUSA, 2006).

O último parâmetro a ser configurado é o VPBDIV, esse é responsável por definir a velocidade do barramento VPB, onde estão conectados os periféricos. Esse registrador pode assumir os valores: 1, 2 ou 4. Se for empregado 2, o *clock* do barramento VPB será a metade do *clock* do processador (SOUSA, 2006).

7.3 Comunicação serial

A comunicação serial é importante no projeto, uma vez que é através dela que ocorrerá a comunicação entre o receptor GPS e a placa de desenvolvimento. A primeira etapa na

comunicação serial está na configuração da UART, lembrando que a placa de desenvolvimento MBC2130 possui duas UARTs. Na Figura 27, localizada na página 69, pode ser visualizado o código fonte responsável por configurar a UART1 e a sua interrupção para o recebimento dos bytes.

A primeira variável a ser configurada é a seleção dos pinos. Para tal é atribuído o valor hexa 0x00050000 (1010000000000000000 em binário). Para entender essa seleção de pinos é necessário analisar a Figura 6, localizada na página 23, demonstra a utilização dos pinos no processador ARM7 (ARM, 2004).

A próxima variável definida é U1LCR, utilizada para definir parâmetros da comunicação serial como: tamanho da palavra a ser transmitida ou recebida, número de *stop* bits, paridade, *baud rate*. No projeto ela foi configurada como 0x83 em hexa (10000011 em binário). O bit 0 e 1 são utilizados para representar o tamanho da palavra, sendo que 11 configura para a utilização de 8 bits. O bit dois configura *stop* bits, onde este bit em 0 define 1 *stop* bit e quando for 1 define como 2 *stops* bits. O bit 3 define a utilização ou não de paridade, onde 0 desabilita e 1 habilita. O bit 4 e 5 definem o tipo de paridade, onde 00 (paridade ímpar), 01(paridade par), 10(forçar bit paridade 1) e 11(forçar bit paridade em 0). O bit 6 determina se ocorrerá controle de parada, esse bit configurado como 0, desabilita a parada de transmissão e quando em 1 habilita. O último bit é utilizado para determinar se habilita (bit em 1) ou não (bit em 0) o controle de *baud rate*.

Os registradores U1DLL e U1DLM são utilizados para definir a taxa de comunicação serial. Na comunicação serial entre o receptor GPS e o kit de desenvolvimento desse projeto foi utilizada a taxa de 4800 BPS, pelo fato do receptor GPS vir com essa taxa como padrão. É importante lembrar que os sistemas que irão realizar a troca de informação serialmente, precisam necessariamente estarem utilizando a mesma taxa de comunicação serial.

Para determinar o valor a ser utilizado nos registradores, calcula-se: ((CLOCK_PROCESSADOR / BAUD RATE DESEJADO) / 16). Após se deve colocar a parte inteira da divisão no registrador U1DLL. Caso o valor inteiro for superior a 255, o byte inferior será atribuído ao registrador U1DLL e a parte superior ao registrador U1DLM.

Para definir os valores do *baud rate* é necessário anteriormente habilitar o bit 7 do registrador U1LCR e após definidos os valores dos registradores U1DLL é necessário desabilitar o bit 7 do registrador U1LCR, esse bit é responsável por permitir a escrita nos registradores U1DLL e U1DLM. Na Figura 27 é possível reparar que foi implementada uma função para facilitar essa tarefa.

O registrador U1IER é utilizado para habilitar interrupções de recebimento e transmissão serial. Esse registrador é baseado em três bits, onde o primeiro bit é utilizado para habilitar a interrupção de recepção com o byte recebido no buffer, o segundo bit é utilizado para habilitar a interrupção de transmissão quando o buffer estiver vazio e o terceiro bit habilita a interrupção de recepção de bytes com falha. Para o desenvolvimento do presente projeto foi utilizado somente a interrupção de recepção com o byte em buffer, isso pode ser visualizado na Figura 27, onde o registrador é configurado com o valor 0x01 (001 em binário).

O último passo da inicialização serial é a definição de uma rotina a ser invocada quando ocorrer o recebimento de um byte. Na Figura 27 o código que ajusta esse procedimento é "**VICVectAddr1 = (unsigned long) interruptUart1;",** onde é passado para o vetor de interrupções o identificador da função a ser acionada.

O código "**VICVectCntl1 = 0x20 | 7;**", na Figura 27 é importante e requer atenção no momento de ser configurado. Existem 16 *slots* que são definidos diretamente na variável de controle VICVectCntlX, onde X assume valores entre 0 e 15. Cada *slot* é capaz de ter 32 canais e cada um desses é relacionado a um periférico. A determinação do canal é dado pelo bit 0 ao 4, o bit 5 determina se o *slot* será habilitado. É importante lembrar que, quanto menor o *slot* utilizado, maior será a prioridade da interrupção.

Reparando na configuração realizada na Figura 27, percebe-se que o *slot* 1 foi configurado com o valor 27 em hexadecimal (100111 em binário). Os bits de 0 a 4 determinam o canal, sendo esse valor 7, o que representa o canal da UART1, o bit 5 em 1, sinaliza que o canal está habilitado (SOUSA, 2006).

O Registrador "**VICIntEnable**" é destinado para a ativação de uma interrupção em relação a um periférico. Para ativar a interrupção relacionada a UART1, basta ativar o bit 7.

No código apresentado na Figura 27 utiliza o deslocamento de bits para realizar essa tarefa, porém, poderia ter sido configurado com o valor de 0x80.

```
void init_uart1 (void) {

  PINSEL0 |= 0x00050000;
  U1LCR   = 0x83;
  com_baudrate (4800, 1);
  U1LCR   = 0x03;
  U1IER   = 0x01;
  U1FCR   = 0x07;

  VICVectAddr1 = (unsigned long)interruptUart1;
  VICVectCntl1 = 0x20 | 7;
  VICIntEnable = 1 << 7;
}
void com_baudrate (unsigned int baudrate, int selectSerial) {
  unsigned long DLreload;

  DLreload = (30000000UL / baudrate) / 16UL;

  if(selectSerial == 0){
     U0DLL = DLreload;
     U0DLM = (DLreload >> 8);
  }

  if(selectSerial == 1){
     U1DLL = DLreload;
     U1DLM = (DLreload >> 8);
  }
```

Figura 27 - Código fonte de inicialização serial
Fonte: do autor

Após a configuração da interrupção e da inicialização serial, é necessário determinar as rotinas responsáveis pelo envio e recebimento dos bytes. Na Figura 28 podem ser visualizadas duas rotinas de envio serial, onde a primeira é responsável pelo envio de um byte e a segunda rotina é responsável pelo envio de um vetor de bytes. Basicamente as duas rotinas têm um funcionamento idêntico, o importante é a interpretação do registrador U1LSR, esse registrador mantém o status do envio serial. Se o bit 5 estiver configurado com 0, indica que ainda existem bytes a serem transmitidos, e 1 indica que o buffer de transmissão está vazio. Analisando as rotinas da figura 31 observa-se que o sistema fica aguardando o status de 1 no bit 5, para então enviar o byte correspondente, isso é realizado na seguinte linha de comando: "**while (!(U1LSR & 0x20));".**

A rotina “**SinalizaEnvioRxSerial();**” é utilizada para sinalizar o envio de bytes pela serial, toda vez que um byte é transmitido o status do LED 2 é alterado. A troca do status do LED é extremamente rápida, porém é possível perceber oscilações, o suficiente para visualmente verificar que os bytes estão sendo enviados pela serial. A utilização de um timer para uma melhor visualização da transmissão serial acarretaria em um péssimo desempenho na comunicação serial.

O real envio dos dados pela serial acontece na seguinte linha de código: “**U1THR = ch**” . Para tal o registrador U1THR deve receber o byte a ser enviado. Ao se escrever neste registrador, o dado é enviado pela UART1.

```
int putchar (int ch)  {
  while (!(U1LSR & 0x20))
  SinalizaEnvioRxSerial();
  return (U1THR = ch);
}

void SendTxSerialData (void)  {
  int i=0;
  for(i=0;i< ptTxSerialData; i++){
     while (!(U1LSR & 0x20));
     SinalizaEnvioRxSerial();
     U1THR = TxSerialData[i];
  }
}
```

Figura 28 - Código fonte responsável pelo envio serial
Fonte: do autor

A Figura 29 exibe a rotina de recebimento serial, essa função é chamada sempre que uma interrupção serial de recebimento ocorre na UART1. A primeira tarefa a ser realizada após o recebimento serial é a verificação do status do recebimento. Tal procedimento é realizado através do registrador U1IIR. O bit 1 ao 3 do registrador U1IIR identificam a ocorrência da interrupção. Os seguintes resultados podem ser obtidos na verificação do registrador U1IIR, sendo eles: 1 (buffer de transmissão vazio), 2 (dado disponível no buffer de recepção), 3 (falha na recepção do byte), 6 (indicador de *timeout*).

É importante observar que os procedimentos de recebimento serial somente são processados se na verificação do status o mesmo conter dados disponíveis no buffer. Uma vez

se confirmando, adiciona-se o byte recebido no vetor de recepção e sinaliza-se o recebimento de uma byte para a máquina de estados do programa principal. Nesse momento há chamada da função chamada "**SinalizaRecebimentoRxSerial();**", isso faz com que ocorra a troca do status do LED 1, indicando que estão sendo recebidos bytes.

A última linha de código presente na função da Figura 29 é obrigatória na utilização de interrupções. Tal atribuição torna o *flag* da interrupção inicializado.

```
void interruptUart1 (void) __irq  {

  int statusIIR=0;

  statusIIR = (U1IIR & 0x0E) >> 1;

  if( statusIIR == 0x02 ){

     RxSerialData[ptRxSerialData] = U1RBR;
     ptRxSerialData++;
     MaqEstadosRxSerial           = 0x01;
     SinalizaRecebimentoRxSerial();
  }

  VICVectAddr = 0x00000000;
}
```

Figura 29 - Rotina de recebimento serial
Fonte: do autor

A comunicação serial está configurada para realizar comunicação entre o computador e o kit de desenvolvimento Keil. Contudo, o receptor GPS utiliza o padrão serial TTL, acarretando um problema para o presente projeto. A Tabela 12 explicita a diferença entre os dois padrões. No padrão serial TTL são utilizados 5 Vcc para representar o nível lógico 1 e 0 Vcc para representar o nível lógico 0, enquanto isso o padrão RS232 utiliza o intervalo de -3V a -18V para representar o nível lógico 1 e 3V a 18V para representar o nível lógico 0 (SOUZA, 2008).

Tabela 12 - Comparativo entre TTL e RS232

Nível Lógico	TTL	RS232
1	5 Vcc	-3V a -18V
0	0 Vcc	3V a 18V

Fonte: Artigo desenvolvido por Vitor Amadeu Souza

Na Figura 30 nota-se a diferença entre os padrões seriais acompanhando o valor do sinal dentro do espaço de tempo. É possível reparar que está sendo transmitido o seguinte valor binário: 01101010. Uma das grandes diferenças entre os dois padrões é a distância em que podem ser utilizados. Enquanto o padrão RS232 pode chegar a distâncias acima de 30 metros, o padrão serial TTL é desenvolvido para curtas distâncias, geralmente utilizado para comunicações entre processador e sensores (CEREBELUMM, 2008).

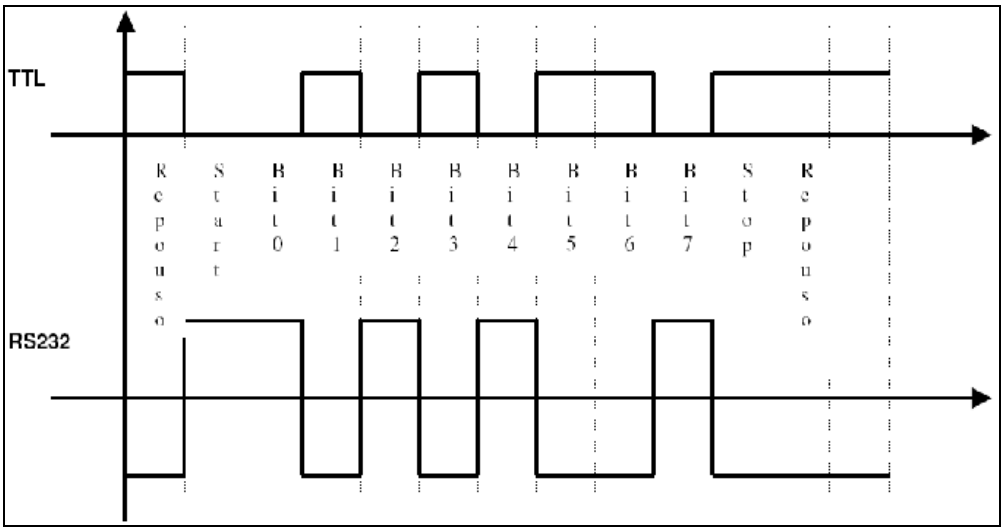

Figura 30 - Exemplo de transmissão TTL e RS232
Fonte: site http://www.cerne-tec.com.br/Artigo_12.pdf

Para solucionar essa diferença na comunicação serial, foi desenvolvido um conversor serial, que transforma o sinal serial TTL em serial RS232. Esse conversor foi fabricado por Fabiano Horn, formado em Mecatrônica em Caxias do Sul no Rio Grande do Sul. O novo hardware presente no projeto é composto basicamente de um CI, quatro capacitores e um resistor. A comunicação entre o receptor GPS e o kit de desenvolvimento somente é possível pela presença do conversor serial, que pode ser visualizado na Figura 31.

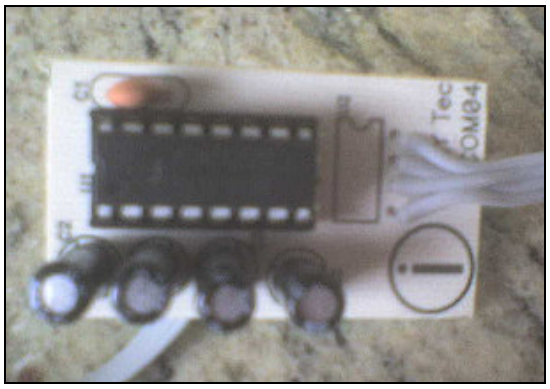

Figura 31 - Conversor serial TTL para serial RS232
Fonte: do autor

7.4 Análise dos dados obtidos

Todos os dados obtidos através do receptor GPS que serão listados a seguir são de origem fidedigna, ou seja, foram obtidos através de monitoramentos realizados em percursos da região. Esses dados serão utilizados para a criação das métricas capazes de classificar o condutor.

Na Figura 32 é apresentada a estrutura física do projeto, onde aparecem números que identificam as partes envolvidas, podendo essas serem identificas como:

1. Receptor GPS;
2. Conversor serial TTL/RS232;
3. Kit de desenvolvimento da Keil MCB2130;
4. Cabo de alimentação do sistema;
5. Cabo serial de comunicação do kit da Keil com o notebook;
6. Cabo da antena do receptor GPS;
7. Notebook;
8. Antena do receptor GPS .

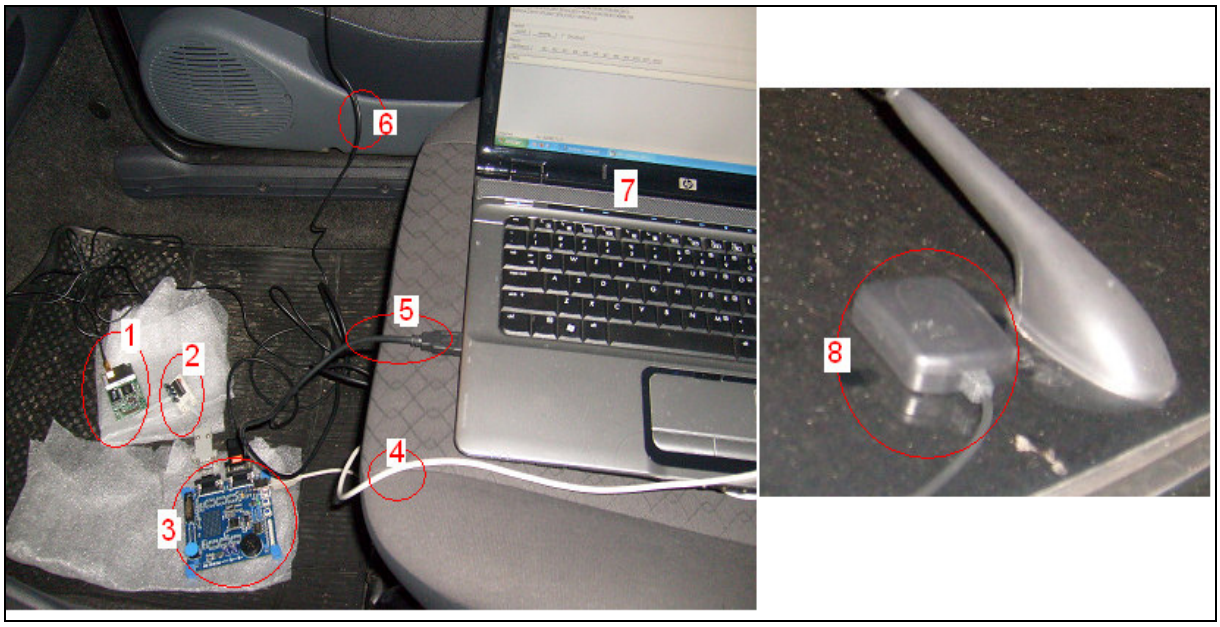

Figura 32 - Estrutura utilizada para capturar os dados do receptor GPS
Fonte: do autor

Para que fosse possível receber as informações vindas do receptor GPS, fez-se necessário o desenvolvimento de uma função no programa do processador ARM. Essa função é responsável por receber as informações na UART0 e retransmití-las para a UART1, que estabelece comunicação com o computador.

O volume de dados recebidos do receptor GPS é de extrema dimensão, para se ter uma mensuração do feito, cita-se o monitoramento realizado entre dois municípios do Estado do Rio Grande do Sul, Venâncio Aires e Santa Cruz do Sul, que representa um trecho de aproximadamente 30 Km. Durante este percurso gerou-se um arquivo texto de aproximadamente 5.070 linhas, totalizando um total de 339 mil caracteres, sendo que o receptor GPS está configurado para receber apenas os protocolos RMC, GSA e GGA.

Para analisar os dados recebidos do receptor GPS e criar as métricas para classificação do motorista foi necessário o desenvolvimento de um software, que realiza diversas operações matemáticas e estatísticas sobre as informações obtidas através do receptor GPS. Para operar o sistema é necessário fazer o uso de dois procedimentos: obter o arquivo gerado a partir do receptor GPS e calcular os dados.

O sistema desenvolvido é exibido na Figura 33. Através desse é possível visualizar graficamente a forma com a qual cada condutor conduz o veículo. Esse sistema permitiu a comprovação matemática diferencial na forma de conduzir de um motorista e outro. No decorrer do trabalho serão exibidos e analisados alguns resultados obtidos.

Para fim de alcançar os objetivos estipulados no projeto, foi necessária a criação de indicadores, sendo eles: limite de velocidade, aceleração, desaceleração e constância e amplitude de velocidades. Esses indicadores terão o intuito de classificar o motorista. Na Figura 33 é possível observar que o sistema disponibiliza todos os indicadores mencionados acima em suas abas.

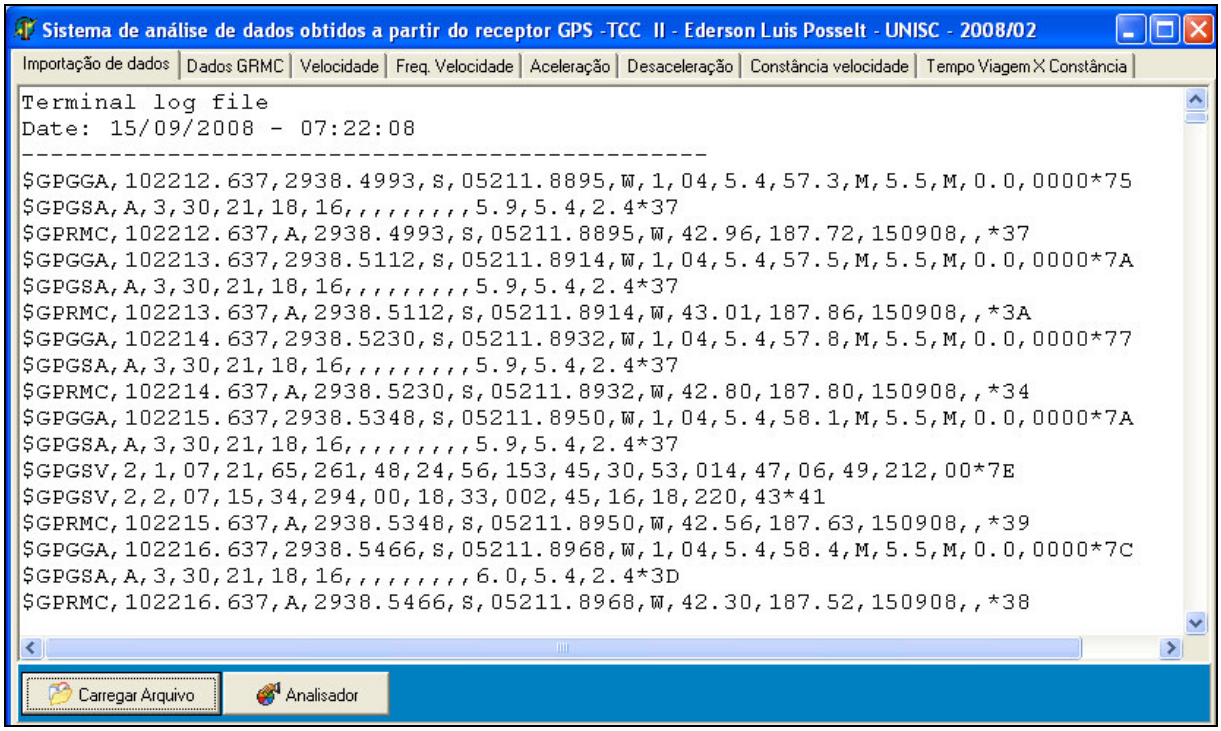

Figura 33 - Sistema de análise dos dados para geração das métricas do sistema
Fonte: do autor

Para melhor entendimento do sistema desenvolvido, serão analisados os devidos indicadores, visualizando os seus respectivos gráficos. A Figura 34 apresenta o comparativo entre dois gráficos: o primeiro exibe dados obtidos de um perímetro rodoviário e o segundo um perímetro urbano. Os gráficos analisados são referentes a velocidade em relação ao tempo.

O primeiro gráfico exibe os resultados obtidos no monitoramento do percurso Venâncio Aires/ Santa Cruz do Sul, sendo facilmente observável uma velocidade elevada e

uma maior constância na velocidade. Um dos fatores importantes é a precisão do receptor GPS, em todos os monitoramentos realizados as informações obtidas em relação ao efetuado eram equivalentes. Uma demonstração desse fato pode ser visualizada no gráfico, onde em torno das 10h29min19s, ocorre uma parada no percurso por um intervalo de aproximadamente 30 segundos, essa parada é ocasionada em decorrência do pedágio existente em Venâncio Aires, próximo aos limites de com Santa Cruz do Sul.

O segundo gráfico demonstra a análise dos dados obtidos em um trecho urbano em Venâncio Aires. Nesse gráfico é possível observar uma velocidade menor, compatível com a de trechos urbanos da região e uma grande inconstância na velocidade, algo característico de ambientes urbanos.

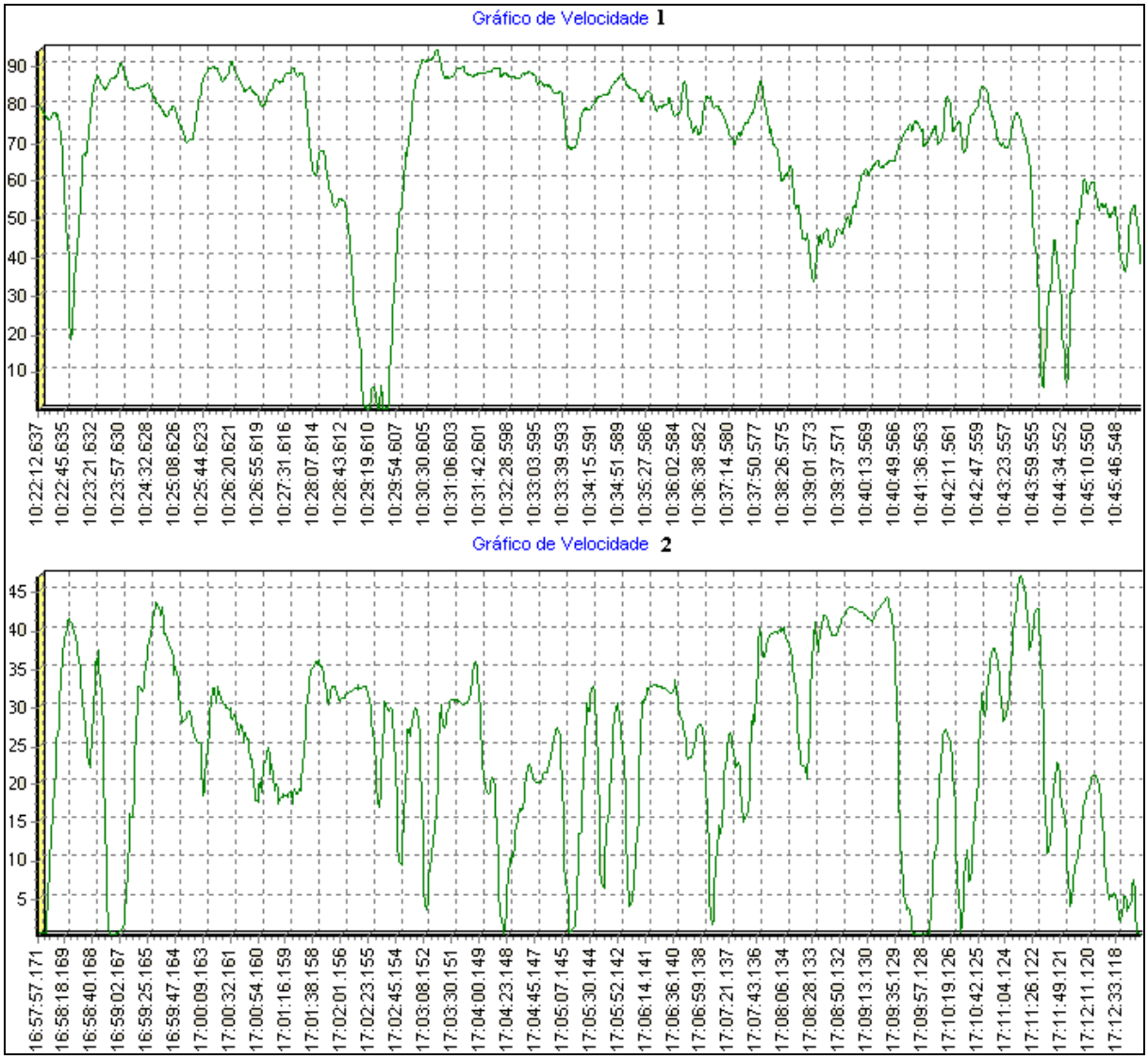

Figura 34 - Gráfico de Velocidade por tempo
Fonte: do autor

Para complementar o gráfico da Figura 34, existe o gráfico da Figura 35 que visa fazer uma distribuição das freqüências das velocidades obtidas no percurso descrito anteriormente, divididas em intervalos de 10 Km/h. Analisando o gráfico de distribuição de freqüências, explicita-se a diferença entre trajetos urbanos e rodoviários.

Estudando os gráficos da Figura 35 pode-se observar que no primeiro gráfico a maior freqüência ficou registrada nas velocidades entre 80 e 90 km/h, representando aproximadamente 38 % do percurso, sendo relativamente distante das freqüências registradas nos outros intervalos de velocidades do mesmo gráfico. A velocidade máxima atingida foi inferior a 100 km/h.

No segundo gráfico pode-se observar a oscilação entre duas freqüências que se mantiveram muito próximas, 20 a 29 km/h e 30 a 39 km/h. Tem-se como esperado que em um trecho urbano composto de muitas paradas e arrancadas se obtenha esse tipo de reconhecimento, outro detalhe importante é que a velocidade máxima foi inferior a 50 km/h.

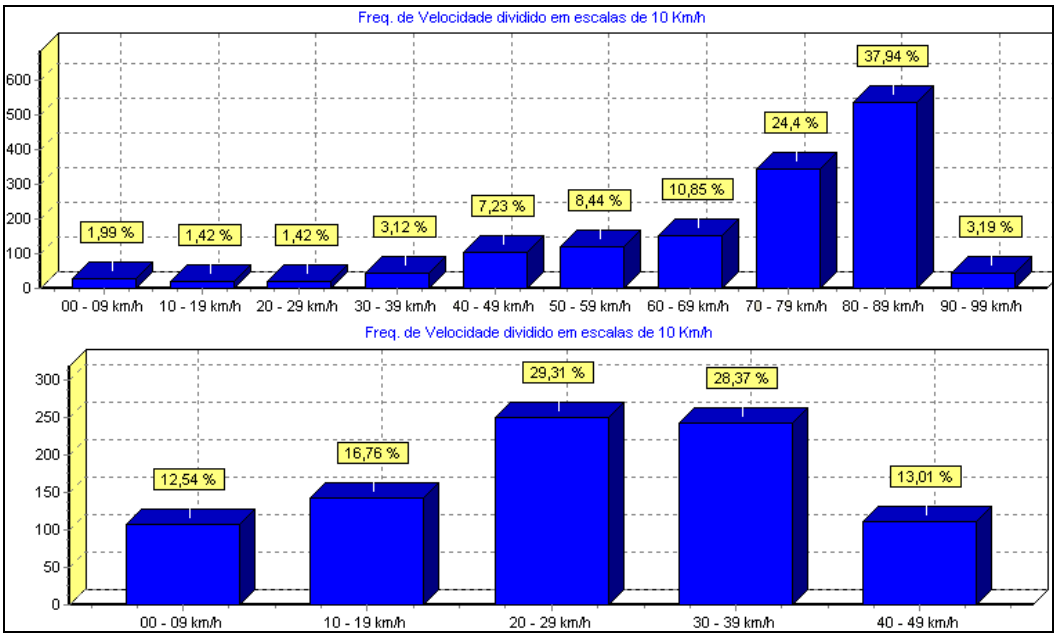

Figura 35 - Gráfico da distribuição das velocidades
Fonte: do autor

Na Figura 36 é possível observar o gráfico de aceleração e desaceleração. Os dados visualizados neste gráfico são referentes aos mesmos dados utilizados nos gráficos apresentados anteriormente. Os gráficos 1 e 2 da Figura 36 são referentes ao percurso realizado na rodovia e os gráficos 3 e 4 são referentes aos realizados no percurso urbano. Foi

considerada como aceleração e desaceleração a diferença entre a velocidade atual e a velocidade anterior, tendo com diferença temporal 1 (um) segundo. Essa diferença de tempo foi empregada porque o receptor GPS garante o envio dos pacotes NMEA neste intervalo de tempo.

Analisando os gráficos, observou-se que o comportamento da aceleração e da desaceleração não apresentou números de grande diferença na forma de aceleração e desaceleração em relação ao tipo de percurso. Essa informação demonstra que a forma com que se aumenta ou diminui a velocidade é semelhante na rodovia e na cidade, logicamente os índices de aceleração e desaceleração serão maiores nos perímetros urbanos, devido ao fato da necessidade de se realizar diversas paradas e arrancadas, porém nada justifica arrancadas que atinjam acelerações acima de 6 km/h.

O ponto mais importante exibido nos gráficos de aceleração e desaceleração é a freqüência em que ocorrem as mesmas. Pode-se reparar que em todos os gráficos o fato comum é a progressão ou redução de 1 Km/h no intervalo de 1 segundo. Fica evidente que quanto mais concentrada for a freqüência relacionada a 1 km/h, mais confortável e segura será a viagem, uma vez que não são registradas acelerações e freadas bruscas.

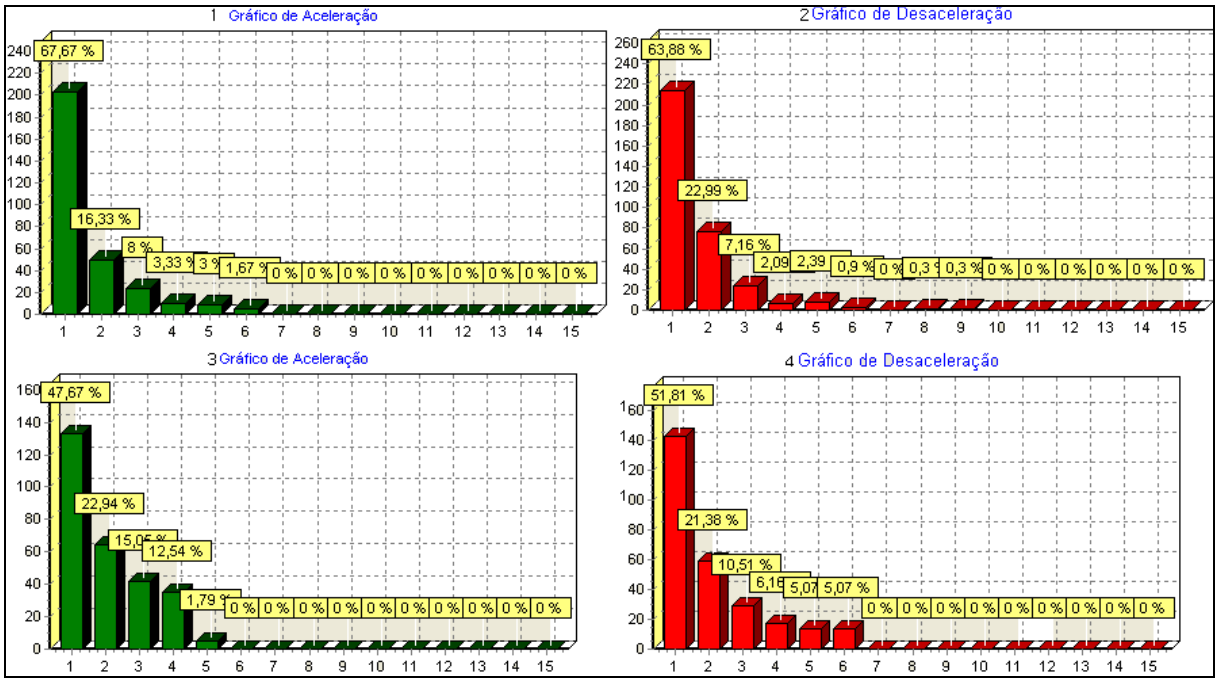

Figura 36 - Gráfico de aceleração e desaceleração
Fonte: do autor

Observando os gráficos acima, na Figura 36, fica a impressão de que seja impossível visualizar dados que tenham uma variação considerável em relação aos demonstrados. Nesse momento o presente projeto consegue comprovar que, se um determinado motorista se tornar um “piloto”, ele não passará despercebido pelo sistema. Para comprovar essa afirmação basta visualizar e analisar a Figura 37, que mostra que a maior freqüência passou de 1 km/h por segundo para 4 km/h em um segundo, o que representa que as acelerações foram pelo menos 4 vez mais intensas.

Os dados gerados no gráfico da Figura 37 foram obtidos em um percurso que não oferecia perigo, uma vez que o veículo foi acelerado com mais intensidade em relação a forma padrão de dirigir. Considera-se como forma padrão de dirigir, as normas estabelecidas para a obtenção da CNH (Carteira Nacional de Habilitação). Uma informação importante é a de que essas acelerações foram obtidas com um veículo da FIAT com motor de 1000 cilindradas, sem forçar muito a aceleração do veículo. Deixando a imaginação fluir, imagine esse gráfico sendo realizado por um automóvel com um motor de 2000 cilindradas e no seu volante um motorista que acredita ser um piloto nas ruas da cidade.

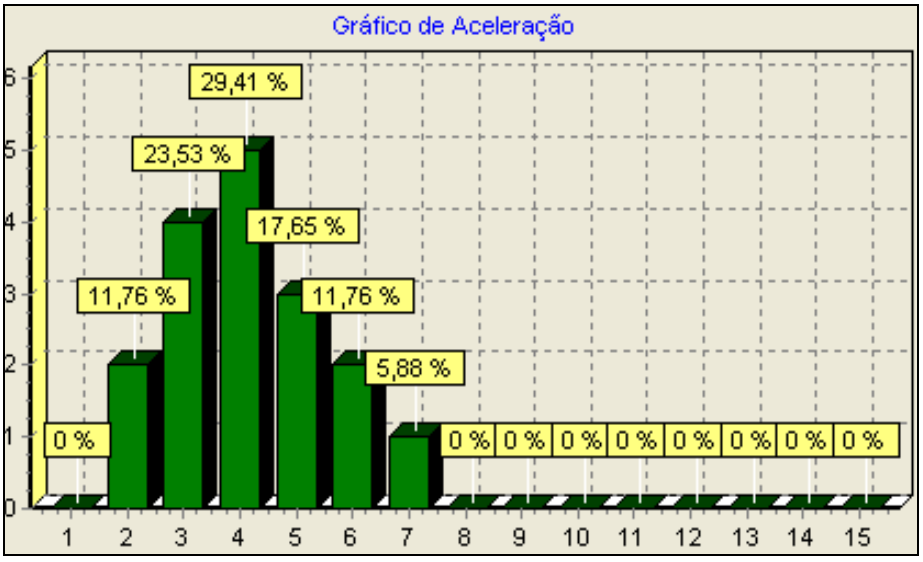

Figura 37 - Gráfico de aceleração agressiva.
Fonte: do autor

Na Figura 38 é exibido o gráfico de constância de velocidade obtida pelo motorista em um determinado trajeto. As regras adotadas para obtenção da constância são as seguintes: é necessário permanecer em uma determinada velocidade, podendo variar 9 Km/h para mais ou para menos por um período superior a 30 segundos.

A constância de velocidade representa várias informações importantes em uma viagem:

- Conforto;
- Economia de combustível;
- Menor manutenção do veículo;

Visualizando o gráfico da Figura 38 observa-se que a constância de velocidade obtida no percurso Venâncio Aires/ Santa Cruz do Sul, foram obtidos em vários momentos com constância de velocidade. O maior tempo registrado de constância foi de três minutos e doze segundos a uma velocidade média de 86 Km/h, o que representou um deslocamento de aproximadamente 4,5 Km.

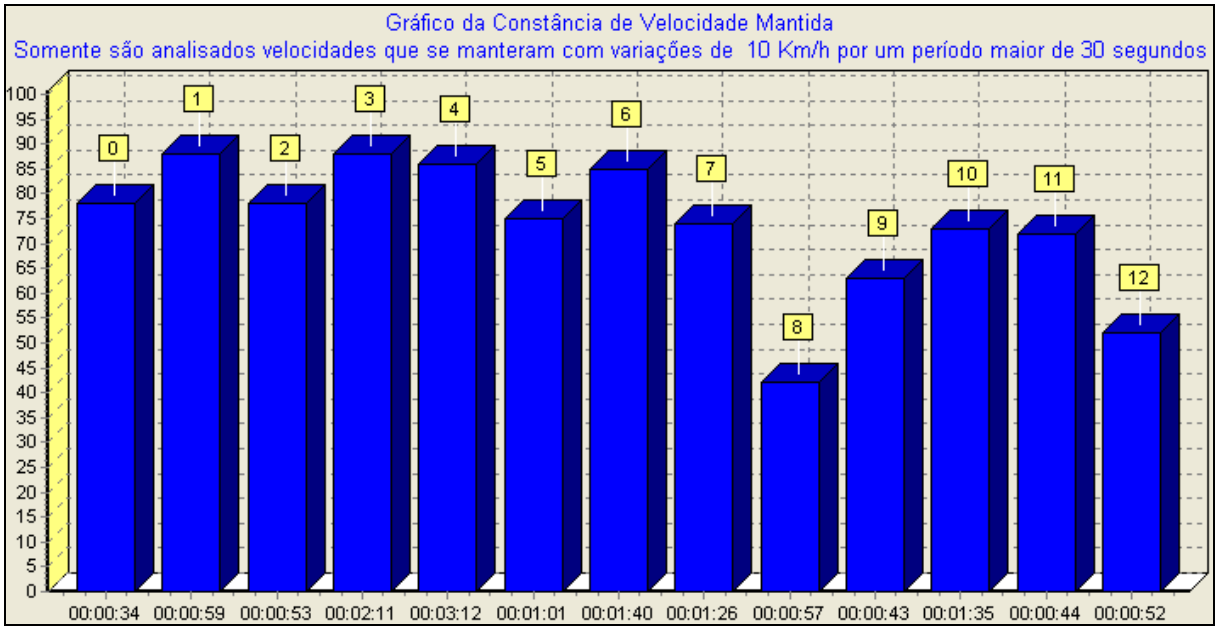

Figura 38 - Gráfico da constância de velocidade
Fonte: do autor

Com o gráfico de constância podem-se extrair várias conclusões. Na figura 39 são exibidos dois gráficos de constância de velocidade totalizados em relação ao tempo total de viagem, o primeiro em percurso rodoviário e o segundo em percurso urbano. Para entender esse gráfico é necessário observar as das colunas, a coluna central representa o total do tempo de viagem, a coluna da esquerda representa a soma do tempo de velocidade constante obtida do trajeto e a coluna da direita sinaliza o total de tempo que o veículo permaneceu parado.

A diferença entre os gráficos ocorre em função do tipo de trajeto. Pode-se observar que no percurso rodoviário obteve-se aproximadamente 70% de constância na velocidade, enquanto que no trajeto urbano o índice foi de apenas 39%. Outra variável que se salientou foi a do tempo em que o veículo permaneceu inativo, no trajeto rodoviário o total de tempo parado foi de 21 segundos, decorrentes do tempo de efetuar o pagamento da tarifa do pedágio, enquanto isso, no percurso urbano obteve-se 43 segundos com o veiculo parado, em decorrência de cruzamentos e sinaleiras.

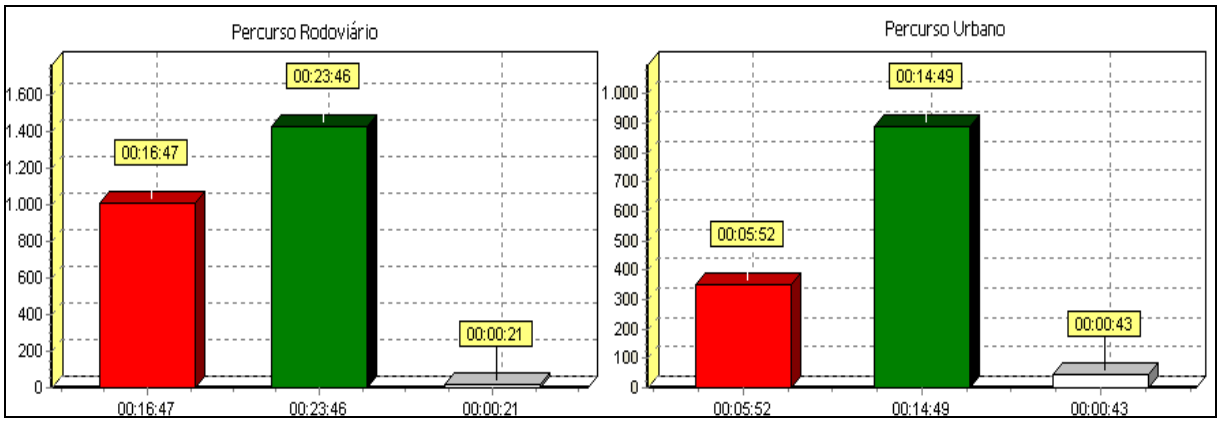

Figura 39 - Gráfico comparativo entre a constância em percurso rodoviário e urbano
Fonte: do autor

Nas Tabelas 13, 14 e 15 abaixo, é possível realizar análises estatísticas de diferentes percursos: rodoviário, secundário e urbano. Essa informação pode ser importante na classificação do condutor, na medida em que, podem-se ter regras diferentes conforme o trajeto realizado.

Considerando o percurso urbano exibido na Tabela 13, pode-se reparar nas seguintes características: a velocidade máxima não ultrapassou os 50 Km/h e a média de velocidade ficou entre 15 e 25 Km/h. Esses valores são normais para regiões urbanas, onde na sua maioria a velocidade máxima permitida é de 40 Km/h, com exceção de vias expressas e anéis rodoviários. A média de velocidade baixa se explica pela quantidade de sinaleiras, esquinas e um trânsito mais intenso. Para a utilização de dados, foram coletadas 5 amostras de dados de trajetos urbanos, onde na sua média aritmética individual, obteve-se uma média de 21,52 Km/h.

Tabela 13- Análise dos percursos urbanos quanto a média e máxima de velocidade.

	Média Velocidade	Velocidade máxima
Análise 1	20,28	40,82
Análise 2	21,76	35,94
Análise 3	24,31	47,17
Análise 4	16,73	37,49
Análise 5	24,52	42,52
Média	21,52	

Fonte: do autor.

A Tabela 14 exibe duas amostras de dados coletados em trajetos secundários, onde geralmente tem-se um limite de 60 Km/h, esse limite pode variar em função dos contratempos da estrada. Pode-se reparar que a velocidade máxima atingida foi de 64,54 Km/h, e a média dos dois percursos foi de 43,95 Km/h. Esses valores são facilmente explicados pelo tipo de rodovia, além de permitir uma velocidade de 20 km/h superior ao trânsito urbano, as estradas secundárias possuem poucos obstáculos que sujeita o motorista a reduzir a velocidade. Os obstáculos mais encontrados são: lombadas, geralmente próximas a escolas ou locais de fluxo de pessoas, curvas acentuadas e estradas estreitas.

Tabela 14 - Análise dos percursos secundários quanto a média e máxima de velocidade.

	Média Velocidade	Velocidade máxima
Análise 1	48,00	64,54
Análise 2	39,19	64,38
Média	43,95	

Fonte: do autor

Na Tabela 15 é cabível visualizar quatro amostras de dados obtidas em rodovias, pode-se observar que a velocidade máxima atingida foi de 100,93 Km/h e que a média das médias das amostras foi de 74,54 km/h. Esses valores são facilmente explicáveis, nas rodovias existe uma estrutura que na maioria dos casos possui um limite de velocidade fixado em 80 Km/h, não havendo muitos obstáculos que obriguem a reduções de velocidade em pequenos espaços. Os tipos de obstáculos que costumam obrigar o condutor a reduzir a velocidade em rodovias são: pedágios, onde é necessária a parada para pagamento e reduções obrigatórias em rótulas, que na maioria das vezes ligam vias secundárias a rodovia, algo muito comum na região da

execução do presente projeto. As análises 1 e 2 foram obtidas no percurso Venâncio Aires/ Santa Cruz do Sul e as análises 3 e 4 foram obtidas no percurso Venâncio Aires/ Porto Alegre. Nesses dois conjuntos é possível observar uma diferença significativa entre as médias de velocidade, porém as velocidades máximas atingidas são muito semelhantes.

Tabela 15 - Análise dos percursos rodoviários quanto a média e máxima de velocidade.

	Média Velocidade	Velocidade máxima
Análise 1	69,29	94,23
Análise 2	58,62	92,38
Análise 3	82,09	96,07
Análise 4	88,14	100,93
Média	74,54	

Fonte: do autor

Com os dados apresentados nas Tabelas 13, 14 e 15 foi possível observar as diferenças de comportamento nos distintos percursos. Mesmo sem um mapeamento da região salvo no equipamento, é possível determinar o tipo de percurso realizado pelo motorista, porém não é possível estabelecer os limites de velocidade do percurso.

Existe uma divisão nítida entre os três tipos de percurso analisados. As diferenças são determinadas pela média de velocidade e pela velocidade máxima. Essas observações são fundamentais no processo de criação das regras de classificação que irão determinar o conceito do condutor.

7.5 Processo de classificação do condutor

Realizar uma classificação, sempre é um processo complicado, que exige muita cautela e responsabilidade, ainda mais quando a classificação é referente ao ser humano. Se analisar-se cautelosamente as informações demonstradas até o momento, não há como definir uma classificação ou perfil para um determinado motorista. Como será possível definir o quanto um motorista é bom ou ruim?

Como forma de validar o projeto, contou-se com o apoio do docente do curso de psicologia da UNISC, Marcus Vinicius Castro Witczak, que munido de todo um conhecimento nesta ciência de estudo, fomentou discussões, fazendo-nos refletir sobre a real necessidade de se obter através de dados, a perfilação referente a de um motorista padrão.

Na tentativa de estabelecer os valores padrões de um motorista, foi necessário aprofundar-se na área estatística e fazer uso da função de "MODA". A informação retornada pela moda de uma série de valores, é o valor que ocorre com maior freqüência. Outra função estatística a ser utilizada no projeto foi a média aritmética, que tem como principal atribuição a determinação do ponto de equilíbrio em uma determinada quantidade de valores (DOWNING; CLARK, 2002).

Baseando-se nas estatísticas apresentadas acima, é possível perceber que: após realizar as duas funções estatísticas, obtêm-se valores importantes, que serão empregados na definição/ classificação do condutor. Sabendo-se que esses valores são referentes a uma forma padrão de dirigir, supõe-se que os indivíduos que se afastarem dessas métricas, estão se afastando da forma considerada padrão de dirigir.

Se for exemplificado essa informação com dados pode-se chegar as seguintes conclusões: considerando a média de aceleração obtida através de várias coletas de dados seja 1,50 km/h e que a moda seja 1 km/h, esses são considerados os valores padrões no ato de conduzir o veículo. Supondo que, ao submeter um determinado motorista aos testes, ele obtenha os seguintes resultados: aceleração média de 5 km/h e a moda igual a 3 km/h. Fica evidente que esse motorista se afastou do padrão, e que provavelmente estava sujeito a maiores riscos no trânsito. O tratamento da imprecisão dos dados para a classificação fica concentrado no controlador *fuzzy*, que contempla os procedimentos para esse tipo de informação.

O processo de formatação de um padrão de motorista assume papel importante no processo da criação de um controlador *fuzzy*, principalmente na etapa de definir as respectivas pertinências e no momento de criar os conjuntos de regras na etapa de inferência.

Após estudos, constatou-se que a utilização da estatística que atua na área da probabilidade, como a distribuição normal, também conhecida como curva de Gauss, não tem

um bom retorno no processo de classificação de um condutor, porém ela abre espaços para novos projetos que visem a utilização do presente projeto.

7.6 Definições do algoritmo *fuzzy* na classificação do condutor

Para ser possível a classificação do condutor através do algoritmo de *fuzzy* baseado nos critérios de classificação mencionados anteriormente é necessário algumas definições referentes ao processo de classificação, entre elas destaca-se:

- Definição das entradas;
- Definição das variáveis lingüísticas e os termos lingüísticos;
- Definir a função e graus de pertinência;
- Construir a base das regras;
- Definir as variáveis de saída.

As variáveis de entrada foram obtidas na análise dos dados do receptor GPS em um intervalo de tempo equivalente a 1 minuto. Utilizou-se essa medida de tempo por ela apresentar as informações de forma mais coerente as situações geradas pelo condutor no trânsito. Seria inviável executar o controlador *fuzzy* a cada segundo, e também pelo fato do controlador *fuzzy* ser alimentado por médias nesse intervalo. As entradas no algoritmo serão: maior velocidade, amplitude total da velocidade, média da velocidade, média de aceleração, maior aceleração, média de desaceleração, maior desaceleração e classificação do condutor.

Para se chegar a essas conclusões, foi necessário realizar uma implementação de um módulo que permitisse visualizar a informação das entradas do sistema *fuzzy* , tais podem ser visualizadas na Figura 40. Após realizar essa implementação, foi possível ampliar as expectativas do presente projeto, abaixo serão listadas as novas considerações obtidas:

Quanto mais a amplitude total da velocidade se aproxima da maior velocidade, maiores serão as acelerações e desacelerações. Isso se deve ao fato do condutor estar realizando uma arrancada ou uma parada. Analisando a Figura 40, na linha de minuto igual a 001, pode-se reparar que a amplitude total da velocidade foi de 43,81 km/h, o que representa aproximadamente 73% da maior velocidade que foi de 75,83 km/h. Nesse momento pode-se

reparar que as acelerações e desacelerações se mostraram maiores do que em momentos em que a amplitude total é pequena. A linha do minuto 002 exemplifica esse caso.

Quanto mais próxima for a média de velocidade da maior velocidade, maior será a constância de velocidade e mais baixas serão as oscilações nas acelerações e desacelerações, isso pode ser constatado em várias linhas da Figura 40, tem-se como exemplo a linha de minuto 002, onde se obtêm uma amplitude de velocidade de apenas 9, 16 km/h e as medidas de aceleração e desaceleração não passaram de 1 km/h.

Quanto mais próximas forem as informações de média de aceleração e maior aceleração, com mais precisão pode ser classificado o motorista, pois essa informação deixa evidente que durante o período analisado o condutor manteve as mesmas características. Quando a média de aceleração for pequena e a maior aceleração muito alta, pode ser um possível erro na informação obtida pelo receptor GPS ou simplesmente um momento em que o motorista teve de realizar uma aceleração maior devido a algum motivo no percurso. Essa regra da aceleração pode ser adotada nas variáveis de desaceleração.

Minuto	Maior Veloc.	Menor Veloc.	Amplitude Total Veloc.	Média Veloc.	Média Aceleração	Maior Aceleração	Média Desaceleração	Maior Desaceleração
001	75,83	32,02	43,81	58,14	0,88	2,09	0,87	1,44
002	85,78	76,63	9,16	83,00	0,32	0,83	0,15	0,41
003	87,28	82,07	5,22	85,95	0,19	0,63	0,30	0,70
004	90,76	58,77	31,99	73,05	0,69	1,39	2,24	6,12
005	92,74	85,90	6,84	88,82	0,35	0,76	0,23	1,05
006	92,28	85,23	7,05	88,82	0,22	0,63	0,34	0,59
007	87,82	85,45	2,37	87,14	0,10	0,28	0,08	0,22
008	87,45	79,96	7,49	82,75	0,05	0,15	0,20	0,46
009	88,60	79,20	9,40	84,06	0,35	0,81	0,21	0,59
010	84,88	82,57	2,31	84,02	0,10	0,28	0,13	0,48
011	82,90	72,45	10,45	79,72	0,05	0,07	0,22	0,96
012	90,19	71,95	18,24	82,92	0,47	1,13	0,33	0,76
013	90,96	84,66	6,31	88,44	0,32	0,67	0,22	0,52
014	90,06	82,60	7,46	86,00	0,21	0,46	0,29	0,52
015	90,22	83,01	7,22	86,45	0,22	0,48	0,45	0,78
016	88,56	83,66	4,90	86,77	0,11	0,28	0,17	0,41
017	87,97	77,79	10,17	84,16	0,24	0,39	0,30	0,72
018	84,92	77,94	6,97	81,18	0,25	0,50	0,14	0,70
019	86,03	73,85	12,17	80,58	0,24	0,46	0,57	1,78

Figura 40 - Entradas do sistema fuzzy
Fonte: do autor.

As variáveis lingüísticas e os termos lingüísticos utilizados são:

- Aceleração: pequena, média, alta, muito alta
- Velocidade: devagar, médio, rápido, muito rápido
- Desaceleração: pequena, média, alta, muito alta
- Classificação do condutor: ruim, regular, bom, muito bom

Após definidas as variáveis lingüísticas e os termos lingüísticos, é o momento de definir as funções e os graus de pertinência. As imagens abaixo apresentam as funções de pertinência das variáveis lingüísticas, adotadas no projeto de classificação do motorista.

A Figura 41 apresenta o diagrama Hassi-Euler (H-E) dos conjuntos *fuzzy* da variável aceleração. É possível reparar que os graus de pertinência variam na escala de 0 até 1 para qualquer entrada no universo de discussão de "x". Exemplificando a citação anterior, afirma-se que o grau de pertinência para a aceleração 8 km/h na Figura 41 tem grau de pertinência igual a 1.

A representação de um conjunto *fuzzy* não precisa necessariamente ser representada por um diagrama, como o da Figura 41, esse poderia ser apresentado pela seguinte seqüência: $\mu_{AP}(X)$ = {1/1, 0,5/2 , 0/3, 0/4, 0/5, 0/6, 0/7, 0/8, 0/9, 0/10, 0/11, 0/12, 0/13, 0/14, 0/15,}, onde informa-se os graus de pertinência para cada entrada no universo de "x". Essa técnica somente pode ser utilizada quando o universo a ser descrito for pequeno. No caso do presente projeto, essa metodologia não se aplica, pelo fato da aceleração poder assumir qualquer valor real no universo do discurso, com duas casas decimais.

Para complementar o diagrama de Hassi-Euler, é importante definir as funções de pertinência para os termos lingüísticos da variável. Na Tabela 16 são exibidas as funções de pertinência em relação a variável de aceleração. É possível reparar que em determinados momentos é exibida a seguinte função: "X-A / B-A". Essa função é responsável por calcular a pertinência de um determinado valor para X, quando o mesmo se encontra presente em uma função de reta linear, que pode ser encontrada no momento em que um termo lingüístico passa do grau de pertinência 0 até 1. É importante criar a tabela de funções de pertinência, pois ela facilita a implementação das rotinas que determinam a pertinência de um elemento.

Tabela 16 - Funções de pertinência da variável de aceleração

$\mu_{AP}(X) = \{$	1, se para X <= 1 X-A / B-A, se 1 < X < 3 0,se X >= 3
$\mu_{AM}(X) = \{$	0, se X <= 2 ou X >= 6 X-A / B-A, se 2 < X <= 4 X-A / B-A, se 4 < X <=6
$\mu_{AA}(X) = \{$	0, se X <= 5 ou X > = 11 X-A / B-A, se 5 < X < 7 1, se 7 <= X <= 9 X-A / B-A, se 9 < X <= 11
$\mu_{AMA}(X) = \{$	0, se X <= 9 X-A / B-A, se 9 <= X < 11 1, se X >= 11

Fonte: do autor.

A Figura 41 exibe as mesmas informações que a Tabela 16, porém de uma forma mais amigável e de fácil compreensão, isso é confirmado facilmente, pois olhando para o gráfico fica fácil reconhecer as variáveis lingüísticas, termos lingüísticos, graus de pertinência e os seus limites.

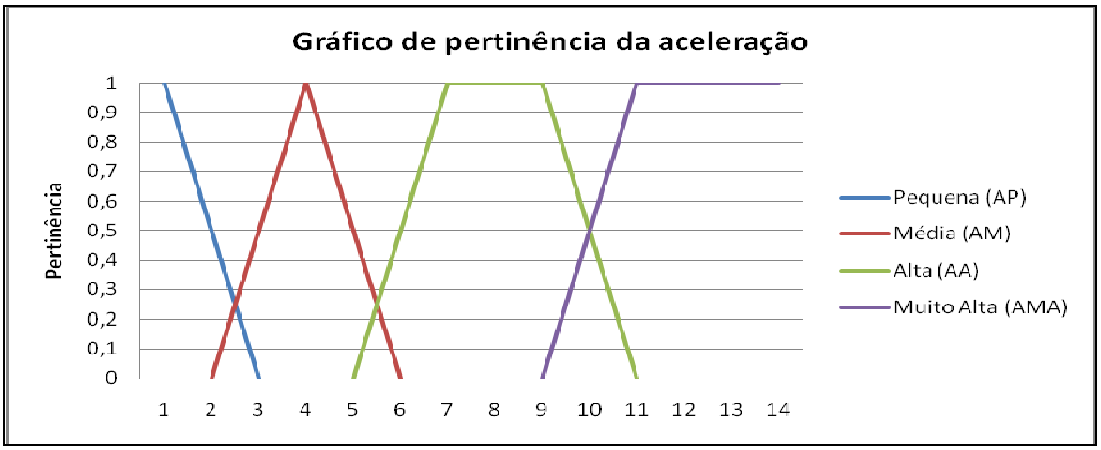

Figura 41 - Diagrama de Hassi-Euler dos conjuntos AP, AM, AA e AMA.
Fonte: do autor

Na seqüência são exibidas as mesmas informações já demonstradas e explicadas acima, porém referentes a variável lingüística velocidade, desaceleração e classificação do motorista. A criação dessas variáveis e a sua análise são fundamentais e indispensáveis para o entendimento do projeto. O sucesso do algoritmo *fuzzy* encontra-se justamente em uma boa definição das variáveis lingüísticas, dos termos lingüísticos e dos respectivos graus de pertinência atribuídos. Logo abaixo, pode-se visualizar na Tabela 17 as funções de pertinência referentes a velocidade, na Figura 42 o diagrama de Hassi-Euler dos termos lingüísticos referentes a velocidade e o mesmo se segue para a desaceleração e classificação do motorista.

Tabela 17 - Funções de pertinência da variável de velocidade

$\mu_{VD}(X) = \{$	1, se para X <= 30
	X-A / B-A, se 30 < X < 50
	0, se X >= 50
$\mu_{VM}(X) = \{$	0, se X <= 40 ou X >= 80
	X-A / B-A, se 40 < X <= 60
	X-A / B-A, se 60 < X < 80
$\mu_{VR}(X) = \{$	0, se X <= 70 ou X >= 110
	X-A / B-A, se 70 < X <= 90
	X-A / B-A, se 90 < X < 110
$\mu_{VMR}(X) = \{$	0, se X < 100
	X-A / B-A, se 100 <= X <= 120
	1, se X > 120

Fonte: do autor.

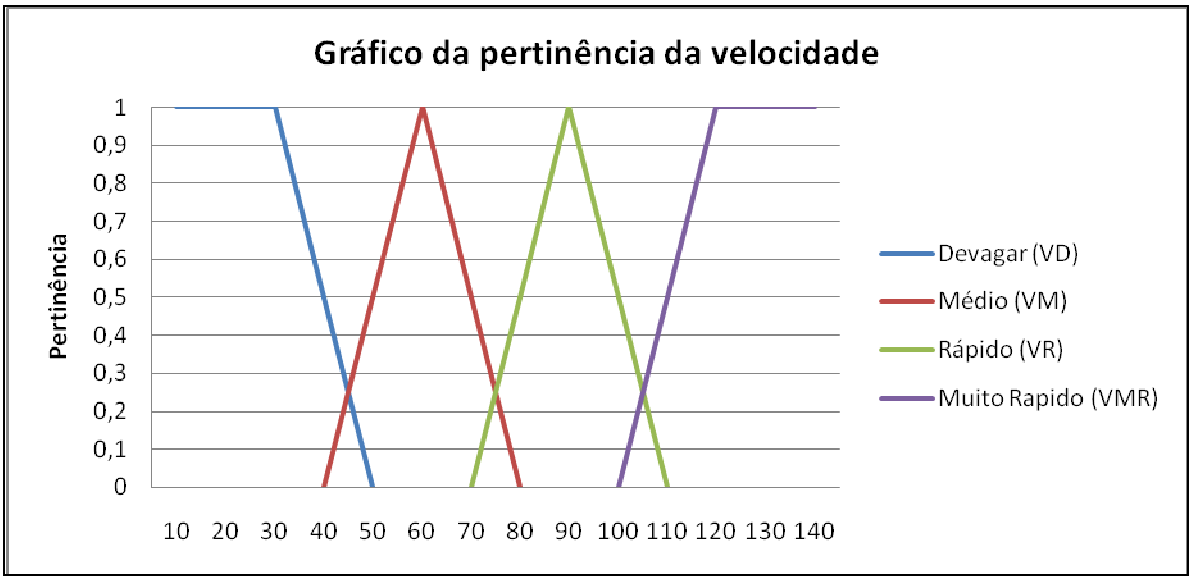

Figura 42 - Diagrama de Hassi-Euler dos conjuntos VD, VM, VR e VMR.
Fonte: do autor

Tabela 18 - Funções de pertinência da variável de desaceleração

$\mu_{DP}(X) = \{$	1, se para X < 1
	X-A / B-A, se 1 <= X <= 3
	0,se X > 3
$\mu_{DM}(X) = \{$	0, se X < 2 ou X > 6
	X-A / B-A, se 2 <= X < 4
	X-A / B-A, se 4 <= X <=6
$\mu_{DA}(X) = \{$	0, se X < 5 ou X > 11
	X-A / B-A, se 5 <= X < 7
	1, se 7 <= X <= 9
	X-A / B-A, se 9 < X <= 11
$\mu_{DMA}(X) = \{$	0, se X < 9
	X-A / B-A, se 9 <= X < 11
	1, se X >= 11

Fonte: do autor.

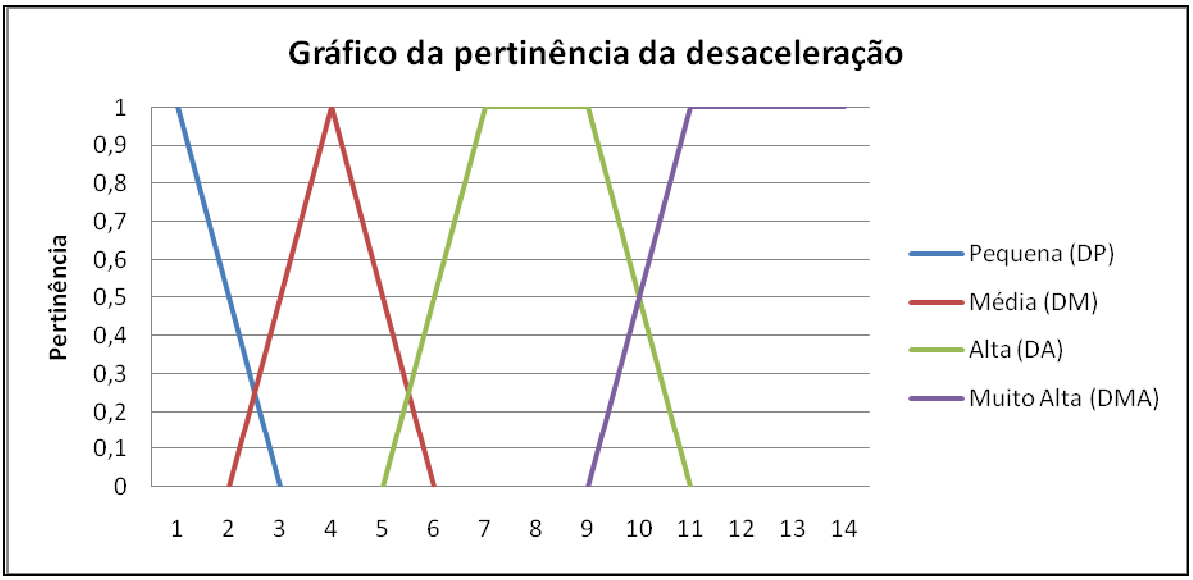

Figura 43 - Diagrama de Hassi-Euler dos conjuntos DP, DM, DA e DMA.
Fonte: do autor.

Tabela 19 - Funções de pertinência da variável de classificação do motorista

$\mu_{AP}(X) = \{$	1, se para $X < 1$
	$X-A / B-A$, se $1 < X < 3$
	0,se $X > 3$
$\mu_{AM}(X) = \{$	0, se $X < 2$ ou $X > 6$
	$X-A / B-A$, se $2 < X < 4$
	$X-A / B-A$, se $4 <= X <= 6$
$\mu_{AA}(X) = \{$	0, se $X < 5$ ou $X > 11$
	$X-A / B-A$, se $5 < X < 7$
	1, se $7 <= X <= 9$
	$X-A / B-A$, se $9 < X < 11$
$\mu_{AMA}(X) = \{$	0, se $X < 9$
	$X-A / B-A$, se $9 < X < 11$
	1, se $X >= 11$

Fonte: do autor.

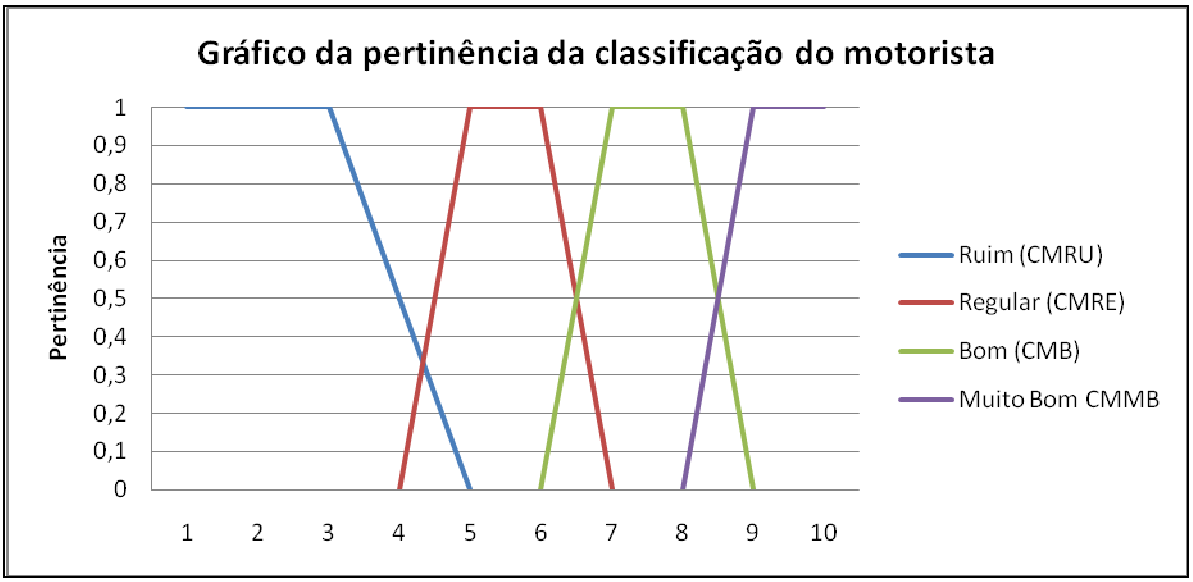

Figura 44 - Diagrama de Hassi-Euler dos conjuntos CMRU, CMRE, CMB e CMMB.
Fonte: do autor

Após as definições das funções e graus de pertinência tem-se o momento de definir as regras de inferência. Pode-se chamar esse ponto como o mais importante do processo, pois é esse o responsável por verificar os antecedentes e gerar as saídas. Analisando a Tabela 20, pode-se verificar o conjunto de regras referentes a maior velocidade e a média da velocidade. A tabela está codificada em siglas que representam os termos lingüísticos, a descrição de cada sigla pode ser visualizada nos diagramas de Hassi-Euler discriminados acima. Utilizou-se essa relação por indicar uma série de fatores que influenciam na classificação do condutor. Para entender melhor a tabela de regras, será dado um exemplo:

Se MaV é VD e MeV é VD **então** CMMB

Traduzindo a regra assim, se obtém: “Se Maior Velocidade é Velocidade Devagar e Média de velocidade é Velocidade Devagar então Classificação Motorista é Muito Bom”. O mesmo vale para o restante dos cruzamentos obtidos na Tabela 20.

Os valores de saída para as definições das regras *fuzzy* foram obtidas com a ajuda do professor Marcus, já mencionando anteriormente e através da análise, possibilitada pelo sistema desenvolvido neste projeto com fins de descoberta de informações nos dados recebidos do receptor GPS.

Em uma análise rápida sobre a Tabela 20 é possível perceber que a medida que a diferença entre a maior velocidade e média de velocidade se acentua, pior será a classificação do motorista. Isso pode ser exemplificado da seguinte forma: sabe-se que médias de velocidade abaixo de 30 Km/h pertencem a percursos urbanos, conforme observado nas amostragens coletadas pelo sistema e tem grau de pertinência igual a 1 no terno lingüístico VD (Velocidade Devagar). Contudo um determinado motorista em um intervalo de tempo, correspondente a 60 segundos, consegue atingir o termo lingüístico VMR (Velocidade Muito Rápida), o que representa uma velocidade acima de 100 Km/h. Isso faz com que o motorista tenha uma classificação ruim.

Porém a medida acima permite uma falha. Suponha que o motorista tenha dirigido por 40 segundos a uma velocidade média de 30 Km/h, e nos últimos 20 segundos, ele tenha elevado a velocidade até 100 Km/h, pelas regras da Tabela 20 o motorista seria classificado possivelmente como ruim, porém, para não cometer nenhuma injustiça, são utilizadas as regras referentes a aceleração e desaceleração. Caso o motorista não tenha excedido esses limites, isso quer dizer, tenha uma boa classificação nas regras referentes a aceleração e desaceleração, a saída *fuzzy* é ajustada e o motorista não é prejudicado. Porém, se o motorista obteve uma classificação ruim nas regras de velocidade e uma classificação ruim na aceleração e desaceleração, o fato está confirmado, o motorista se portou de forma perigosa.

Tabela 20 - Regras fuzzy de velocidade

Maior velocidade					
Média de velocidade		**VD**	**VM**	**VR**	**VMR**
	VD	CMMB	CMB	CMRE	CMRU
	VM		CMMB	CMB	CMRU
	VR			CMMB	CMB
	VMR				CMRU

Fonte: do autor

Na Tabela 21 são apresentadas as regras de aceleração e desaceleração. Ambos os conjuntos de regras, são apresentados em uma única tabela por terem as mesmas regras, sendo assim, o que é válido em relação a aceleração é válido para a desaceleração. As variáveis lingüísticas aceleração e desaceleração são as mais fidedignas. Isso quer dizer que, em qualquer tipo de percurso, o comportamento da aceleração e desaceleração tem que seguir um padrão muito próximo.

Uma característica importante da lógica *fuzzy é* o fato de esta reportar falhas no sistema, isso pode ser visto em uma das regras apresentadas na Tabela 21. Se um individuo possuir uma média de aceleração ou desaceleração pequena, porém no instante de avaliação ele obteve como maior aceleração ou desaceleração a classificação de muito alta, o sistema classificará o motorista como bom, sugerindo que tenha ocorrido algo fora do normal, ou que por motivos externos o motorista, teve que executar uma manobra de segurança, que exigiu por exemplo, uma freada mais brusca.

Tabela 21 - Regras *fuzzy* de aceleração e desaceleração

Maior Aceleração / Maior Desaceleração					
Média de Aceleração / Média de Desaceleração		**AP**	**AM**	**AA**	**AMA**
	AP	CMMB	CMMB	CMB	CMB
	AM		CMRE	CMRE	CMRE
	AA			CMRU	CMRU
	AMA				CMRU

Fonte: do autor

O método utilizado para gerar o conjunto *fuzzy* de saída foi o método min(). Esse método funciona da seguinte forma: sobre todos os antecedentes, utiliza-se o que possuir menor grau de pertinência para atribuir ao conjunto *fuzzy* de saída. Se for utilizado o exemplo apresentado anteriormente tem-se a seguinte saída:

Se MaV é VD e MeV é VD **então** CMMB

Se Mav é 0,30 e MeV é 0,6 **então** μ é 0,30

Foi utilizado o método mi(), por ser mais conhecido, mas existem outros métodos, como prod(), que consiste em atribuir ao conjunto *fuzzy* de saída o produto das duas pertinências (TEIXEIRA, 2001).

Na Figura 45 é possível verificar um exemplo de saída *fuzzy*. É possível reparar que as saídas obtidas foram relativas aos termos lingüísticos “bom” e “muito bom” na variável

classificação do motorista, porém isso não é o suficiente, pois o algoritmos de *fuzzy* recebe dados precisos, converte esses dados para informações nebulosas e por fim obtém um dado preciso de saída. Após aplicar as regras de inferência, obtêm-se conjuntos com as suas respectivas pertinências. É nesse momento em que se deve aplicar o processo de defuzzificação, processo que irá gerar um valor preciso de saída. Como já mencionado anteriormente, existem vários métodos, mas o escolhido para esse projeto é o método do centro da gravidade. Esse método tem como característica, nunca atingir as extremidades, justamente por considerar o centro da área a saída.

A equação para realizar o processo de defuzzificação utilizando o método do centro da gravidade é apresentada abaixo, ela consiste em realizar o somatório da variável multiplicado pelo grau de pertinência da variável dividido pelo somatório do grau de pertinência. Como resultado dessa operação matemática, obtém-se o valor preciso de saída, equação (1).

$$u_{CG} = \frac{\sum_{i=1}^{N} u_i . \mu_u(u_i)}{\sum_{i=1}^{N} \mu_u(u_i)} \qquad \textbf{(1)}$$

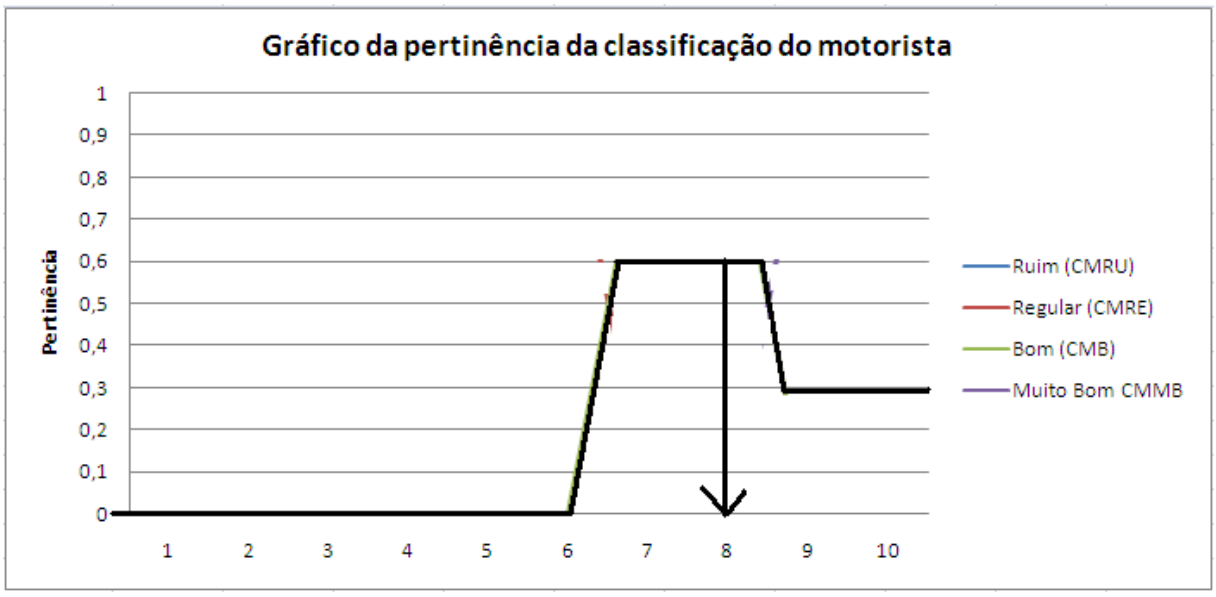

Figura 45 - Gráfico de pertinência da saída fuzzy
Fonte: do autor

Para realmente classificar o motorista, é necessário criar a média do seu comportamento, isso quer dizer, somar todos os resultados obtidos através da saída *fuzzy* e dividir pelo número de ocorrências. Isso irá gerar a nota média do motorista. Isso é necessário, uma vez que o sistema proposto no projeto realiza a execução do controlador *fuzzy* a cada minuto, classificando o motorista. Não adiantaria manter somente a última avaliação, pois existem momentos no trânsito em que freadas e acelerações mais fortes são necessárias, por motivos de segurança. Nesse momento utiliza-se a média aritmética, que gera uma saída capaz de substituir todas as outras, no seu conceito.

7.6 Resultado dos testes realizados

Para legitimar o presente projeto, foram aliciados dois motoristas além do autor, que passarão a serem identificados como motorista A e motorista B, como forma de manter suas privacidades. Os resultados obtidos pelos convidados representaram a realidade com que conduziam o veículo, baseado na relação do motorista padrão.

O trajeto realizado pelo motorista A iniciou na rodoviária de Santa Cruz do Sul, atravessando a cidade até atingir a rodovia. O percurso, tanto urbano quanto rodoviário, totalizou 20 Km rodados; a classificação obtida foi de 8,7 pontos em uma escala de 0 a 10. Em se tratando do motorista B, este efetuou um trajeto semelhante, onde obteve nota 7,8. A nota do autor do projeto foi de 8,5.

Para que fosse possível acompanhar a sua nota enquanto conduzia o veículo, utilizou-se 6 LED`s presentes na placa de desenvolvimento. Na Figura 46, é possível visualizar os 8 LED`s da placa, sendo eles usados da seguinte forma: o primeiro LED sinaliza o recebimento de dados na UART, o segundo LED sinaliza pacotes recebidos do receptor GPS com sucesso e os LED`s de 3 a 8, sinalizam a nota obtida pelo usuário, conforme a Tabela 22.

A visualização através dos LED`s, permite uma noção da nota obtida, entretanto, não exibe com precisão a informação, fato não significativo nesse processo. Saber a faixa de pontuação em que o motorista se encontra, já é o bastante para definir a sua classificação, ou até mesmo gerar um perfil.

Tabela 22 - Esquema da pontuação do condutor

Nota	LED`s Ligados
Entre 9,00 e 10,00	3, 4, 5, 6, 7, 8
Entre 8,00 e 8,99	4, 5, 6, 7, 8
Entre 7,00 e 7,99	5, 6, 7, 8
Entre 6,00 e 6,99	6, 7, 8
Entre 5 e 5,99	7, 8
Menor que 5	8

Fonte: do autor

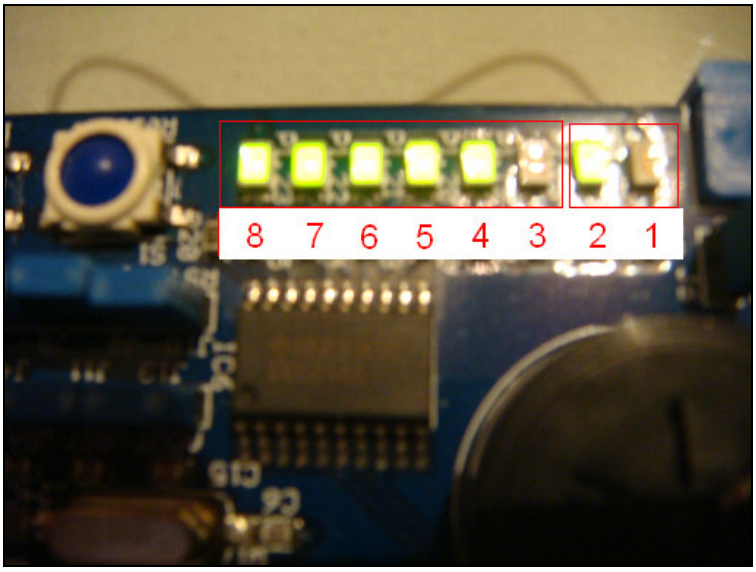

Figura 46 - Visualização da saída *fuzzy* através dos LED`s
Fonte: do autor

O sistema atua de forma pró-ativa, toda vez que o motorista obtém uma saída fuzzy abaixo de 6, através de sinal sonoro, isso perfaz com que o motorista perceba facilmente, que está dirigindo de forma perigosa ou inadequada.

CONCLUSÃO

A qualidade no trânsito nacional vem sofrendo uma degradação acelerada. Um indicador que firma a assertiva mencionada se dá através dos inúmeros acidentes automobilísticos ocorridos diariamente, sendo que na grande maioria desses, a imprudência do condutor ainda permanece como sendo o maior agente destes agravos.

Como forma de atingir o objetivo proposto, foi realizado diversos estudos sobre as tecnologias que permeiam o processador ARM, IDE de desenvolvimento, GPS, sistemas inteligentes, bem como ampliou seu campo de visão em relação à área de psicologia do trânsito, como forma de mapear o tema emergente. É neste sentido que se verifica que a utilização concomitante de hardware específico com o software fomentado pelas teorias de psicólogos da área de trânsito regressa ótimos resultados em relação ao foco principal sugerido.

Com o fechamento dos dados a cerca da implementação do projeto, foi possível atentar para todas as expectativas. Confirmou-se que existem contestações matemáticas na forma de conduzir entre motoristas, e que a classificação no modo pelo qual o motorista conduz o seu veículo é possível. Para este aplicativo utilizou-se a lógica fuzzy. Em todos os testes realizados, a classificação do motorista competia com a forma em que o veículo era conduzido, ratificando que o algoritmo embarcado implementado no processador ARM é consideravelmente hábil para este fim.

O presente projeto pode ser aplicado em diversos ramos de atividade com as mais distintas finalidades. Nas empresas privadas, os veículos somente poderiam ser conduzidos por motoristas que tenham uma avaliação mínima determinada pela empresa. As seguradoras podem estabelecer novos valores de seguros, relativos à classificação do condutor. Outro detalhe a ser lembrado, é o fato de o seguro ter um valor bem elevado para jovens, menores de 25 anos, isso pelo fato das estatísticas constatarem que os jovens tem um envolvimento maior em acidentes, porém, sabe-se que nem todos os jovens tem um perfil agressivo quando no volante de um carro. A utilização do sistema proposto tem condições de reconhecer os jovens que não se enquadram nesse perfil e que poderiam ter um abatimento do valor do seguro.

O sistema proposto também poderia atuar juntamente com os órgãos que controlam o trânsito. Um exemplo seria o uso do projeto pelo DETRAN, onde seria vinculado a renovação da CNH a classificação do condutor, sendo que, motoristas que não atingissem a classificação mínima seriam encaminhados para a reciclagem. Outra utilização poderia ser feita por policiais rodoviários ou de atuação semelhante, onde poderiam deter motoristas que tivessem classificação inferior a exigida, dessa forma se tiraria das ruas os motoristas agressivos.

Outra utilização do projeto seria na área acadêmica. Psicólogos através da utilização do equipamento poderiam realizar várias medições entre públicos diferentes na busca de novos conceitos na área de psicologia do trânsito.

Para tornar o presente projeto mais rígido e confiável na classificação do condutor, seria conveniente que o sistema realizasse a leitura de mais duas variáveis, que seriam: rotação do motor e marcha, pois o sistema atual, não consegue identificar se o motorista utiliza o carro da forma mais adequada. Para exemplificar isso, pode-se utilizar o seguinte caso, um motorista realiza um trajeto de 50 Km em terceira marcha ou em quinta marcha, para o sistema atual a classificação será a mesma para o condutor, porém, o desgaste do carro e o consumo de combustível terão com certeza uma alteração considerável.

Face a sua estruturação, confiabilidade e agilidade, este projeto representará uma importante ferramenta no que diz respeito a segurança e melhoramento da qualidade no trânsito, ampliando horizontes rumo a uma nova linha de pesquisa. Sua ampliação em relação a usabilidade para as mais diversas áreas de estudo é viável e necessária, pois frente a tamanha pluralidade de benefícios, seus dados podem ser integrados a outros sistemas de processamento representando um apoio fundamental em atividades de pesquisa e monitoramento.

Uma boa adequação no equipamento utilizado seria a instalação de um display, esta ação facilitaria a comunicação entre o motorista e o equipamento, outra alteração necessária, seria a adaptação de uma fonte de 12 V, para que se pudesse ligar o equipamento ao acendedor de cigarros do veículo, evitando o uso da alimentação USB.

REFERÊNCIAS

ALVES, Sérgio. *A matemática do GPS.* USP, Revista do professor de matemática, São Paulo, n. 59, p.17-26, Jan/Abr. 2006. Disponível em: <http://www.rpm.org.br/novo/conheca/59/gps.pdf>. Acesso em: maio 2008.

ARM. *ARM7TDMI – Technical Reference Manual,* 2004. Disponível em: <http://www.nxp.com/acrobat_download/datasheets/LPC2131_32_34_36_38_4.pdf>. Acesso em: mar. 2008.

BARROS, A. et al. "Biblioteca para implementação de controladores utilizando lógica fuzzy". Disponível em: <http://www.unifor.br/notitia/file/1087.pdf> . Acesso em: out. 2008

CEREBELUMM. *Serial Communications*. .Disponível em: <http://botrics.com/support/cereb/manual/chapter3/3-5-serial.html>. Acessado em: ago. 2008.

COSTA, A. et al. *Lógica Fuzzy : Conceitos e Aplicações*, Rio Grande do Sul, UNISINOS, Abr. 2005. Disponível em: <http://www.inf.unisinos.br/~cazella/dss/fuzzy_relatorio.pdf>. Acesso em: jun. 2008.

DENATRAN, Departamento Nacional de Trânsito. Disponível em: <http://www.denatran.gov.br>. Acesso em: jan. 2008.

DETRAN-RS, Departamento de Trânsito do Rio Grande do Sul. Disponível em: <http://www.www.detran.rs.gov.br>. Acesso em: fev. 2008.

DOWNING, D.; CLARK, J. *Estatística Aplicada*. São Paulo: Saraiva, 2ª edição. 2002.

FLASH MAGIC, Software para gravar memória flash dos microcontroladores NXP. Disponível em: <http://www.flashmagictool.com>. Acessado em: ago. 2008

GLOBALSAT TECHNOLOGY CORPORATION. User manual GPS Receiver Engine Board: ET102/ER102.

GOMIDE, F. A. C.; GUDWIN, R. R.. Modelagem, Controle, Sistemas e Lógica Fuzzy. Revista SBA Controle & Automação, Rio Grande do Norte, n.3, v.4, set/out. 1994.

GREF: Grupo de Reelaboração do Ensino de Física. *Física I: Mecânica*. São Paulo: EDUSP, 1990.

HOFFMANN, Maria Helena. *Comportamento do condutor e fenômenos psicológicos*. Psicologia: Pesquisa & Trânsito, v. 1, n. 1, p. 17-24, Jul/Dez. 2005. Disponível em: < http://pepsic.bvs-psi.org.br/pdf/ppet/v1n1/v1n1a04.pdf >. Acesso em: jun. 2008.

JÚNIOR, I. N.; NOVAKOWSKI, S. *Lógica Fuzzy Sistemas de Apoio a Decisão*. Rio Grande do Sul, UNISINOS, Abr. 2005. Disponível em: <http://www.inf.unisinos.br/~cazella/dss/SAD_ Logica_Fuzzy.pdf. Acesso em: maio de 2008

KEIL, *Embedded Development Tools*. Disponível em: <http://www.keil.com/mcb2130 2008>. Acesso em: jan. 2008.

KEIL, *MicroVision IDE User`s Guid*. Disponível em: <http://www.keil.com/support/man/docs/uv3/ >. Acesso em: abr. 2008.

KEIL, *MCB2130 User`s Guid*. Keil EUA, 2004

KOSKO, Bart. Fuzzy thinking, the new science of fuzzy logic. New York: Hyperion. 1993

MARIN-LEÓN, L.; VIZZOTO, M. M. *Comportamento no trânsito: um estudo epidemiológico com estudantes universitários*. Caderno de Saúde Pública, Rio de Janeiro, p. 515-523, Mar/Abr, 2003. Disponível em: < http://www.scielo.br/pdf/csp/v19n2/15417.pdf >. Acesso em: jun. 2008.

MONTALVÃO, L.; CARVALHO, N.; *Desenvolvimento de um algoritmo de controle para sistemas de elevadores, com base na lógica fuzzy.* Disponível em: <http://www.frb.br/ciente/2006_2/BCC/CC.MONTALV%C3%83O.etal.F1%20_Rev.%20Vanessa%2024.01.07_.pdf>. Acesso em: out. 2008.

MONTENEGRO, C. A. S.; GÜNTER, H. *Agressividade, raiva e comportamento de motorista*. Psicologia: Pesquisa & Trânsito, v. 2, n. 1, p. 9-17, Jan/Jun. 2006. Disponível em: < http://pepsic.bvs-psi.org.br/pdf/ppet/v2n1/v2n1a03.pdf >. Acesso em: jun. 2008.

MORIMOTO, Carlos E. *Processadores RISC versus processadores CISC*. 2001. Disponível em: <http://www.guiadohardware.net/artigos/risc-cisc/>. Acesso em: abr. 2008.

NETO, L. B. et al. *Minicurso De Sistema Especialista Nebuloso*. XXXVIII Simpósio Brasileiro De Pesquisa Operacional, Goiânia, 2006. Disponível em: < http://www.uff.br/decisao/SBPO%20Fuzy.pdf >. Acesso em: maio 2008.

NMEA: National Marine Electronics Association. *The NMEA 0183 protocolo*, 2001. Disponível em: <http://www.tronico.fi/OH6NT/docs/NMEA0183.pdf>. Acesso em: maio 2008.

NXP, *Homepage do Fabricante do Processador ARM* . Disponível em: <http://www.nxp.com>. Acessado em: Abr. 2008.

PATYRA, M. J.; MLYNEK, D. M.. Fuzzy Logic, Implementation and Applications. New York: Wiley Teubner. 1996.

RENALDI, Felipe. *Genos - Protótipo De Um Montador De Sistemas Operacionais para Sistemas Embarcados*. Santa Catarina: Monografia FURB, 2006. Disponível em: < http://www.bc.furb.br/docs/MO/2006/307628_1_1.pdf >. Acesso em: maio 2008.

REZENDE, Solange Oliveira. Sistemas inteligentes Fundamentos e Aplicações. In: ALMEIDA, P. E. M.; EVSUKOFF, A. G. de (Org.) *Sistemas Fuzzy*. São Paulo: Manole, 2003. p.169-201.

ROCHA, José Antonio M.R. *GPS: uma abordagem prática*. 3. ed. Recife: Bagaço, 2002.

SILVA, L. F.; ANTUNES, V. J. M.. *Comparação entre as arquiteturas de processadores CISC e RISC*. Faculdade de Engenharia da Universidade de Porto, 2007. Disponível em: http://www.inf.ufsc.br/~lucia/Arquivos-INE5607/anterior/Risc-Cisc.pdf>. Acesso em: mar. 2008.

SILVESTRE, E.; BACHIEGA, P. *Estudo sobre processador ARM7*. Santa Catarina: Monografia UFSC, 2007. Disponível em: < http://www.lisha.ufsc.br/~guto/teaching/sys/ine5309-2006-2/work/g4/monografia.pdf>. Acesso em: maio 2008

SOUSA, Daniel Rodrigues. *Microcontroladores ARM7*. São Paulo: Érica. 2006

SOUZA, Vitor Amadeu. *Comunicação em BASIC para o PIC Comunicação Serial* .Disponível em: < http://www.cerne-tec.com.br/Artigo_12.pdf >. Acessado em: ago. 2008.

TANSCHEIT, Ricardo. *Sistemas Fuzzy*. Rio de Janeiro: DEE-PUC-Rio. Disponível em: <http://www.ica.ele.puc-rio.br/cursos/download/LN-Sistemas%20Fuzzy.pdf>. Acesso em: maio 2008.

TEIXEIRA, Rafael Luís. Uma metodologia de projeto de controladores híbridos inteligentes com aplicação no controle ativo de vibrações mecânicas. Uberlândia: Dissertação Mestrado, UFU, 2001. Disponível em: <http://www.bdtd.ufu.br/tde_arquivos/1/TDE-2005-08-12T104641Z-14/Publico/mestrado_rafael_protegido.pdf > . Acessado em: out. 2008.

THIELEN, I. P. et al. *Percepção de risco e velocidade: A lei e os motoristas*. Psicologia: Ciencia e Profissão, Brasília, v.27, n.4, Dez. 2007. Disponível em: <http://pepsic.bvs-psi.org.br/scielo.php?script=sci_arttext&pid=S1414-98932007001200013&lng=pt&nrm=is>. Acesso em: jun. 2008.

ScienceProg. *Configuração do clock do processador ARM7*. Disponível em: <http://winarm.scienceprog.com/arm-mcu-types/clocking-arm-with-crystal-oscillator-and-pll.html>. Acessado em: ago. 2008.

ANEXO A: ENSAIOS REALIZADOS

Nesse anexo serão abordados alguns ensaios realizados sobre os equipamentos disponíveis. A intenção é justamente ampliar o conhecimento sobre os dispositivos conhecendo os seus limites e agregando experiências para desenvolvimentos futuros. Na seqüência seguem os ensaios realizados.

Acionamento de LED´s através de um botão

O primeiro ensaio realizado com a MCB2130 tinha o objetivo de ligar um LED quando o botão INIT1 fosse pressionado. Para dificultar foram utilizados os 8 LED`s presentes na placa MCB2130 e para cada vez que o botão INIT1 fosse pressionado o próximo LED deveria ser ligado e os demais desligados sendo que, quando o LED 8 fosse apagado o LED 1 deveria ligar formando um ciclo. Para incrementar um pouco mais esse ensaio foi sugerido que, se o botão INIT1 se mantivesse pressionado por aproximadamente 1 segundo o próximo LED deveria ser aceso e os demais desligados.

Na Figura 47 é possível visualizar parte do código da biblioteca LPC213x.h, onde estão definidas as constantes com a finalidade de entrada e saída, algumas delas estão presentes no primeiro ensaio. Realizando uma análise detalhada no código fonte exibido na Figura 48, pode-se reparar que na linha um, é feito a inclusão de biblioteca LPC213X.h. Essa biblioteca contém as constantes de manipulação do processador ARM.

```
063 /* General Purpose Input/Output (GPIO) */
064 #define IOPIN0          (*((volatile unsigned long *) 0xE0028000))
065 #define IOSET0          (*((volatile unsigned long *) 0xE0028004))
066 #define IODIR0          (*((volatile unsigned long *) 0xE0028008))
067 #define IOCLR0          (*((volatile unsigned long *) 0xE002800C))
068 #define IOPIN1          (*((volatile unsigned long *) 0xE0028010))
069 #define IOSET1          (*((volatile unsigned long *) 0xE0028014))
070 #define IODIR1          (*((volatile unsigned long *) 0xE0028018))
071 #define IOCLR1          (*((volatile unsigned long *) 0xE002801C))
```

Figura 47 - Parte do código da biblioteca LPC213x.h
Fonte: do autor

```
01 #include <LPC213x.H>
02
03 int main (void) {
04   unsigned int controlLeds = 0x00800000;
05   unsigned int autoIncLeds = 0x001;
06   IODIR1  = 0x00FF0000;
07   while (1) {
08       if ((IOPIN0 & 0x00004000) == 0) {
09
10          autoIncLeds = 0x001;
11          while((IOPIN0 & 0x00004000) == 0){
12             autoIncLeds++;
13             if(autoIncLeds>=0x00E4E1C)
14                 break;
15          }
16
17          if(controlLeds == 0x00800000){
18             controlLeds = 0x00010000;
19             IOCLR1      = 0x00FF0000;
20             IOSET1      = controlLeds;
21          }else{
22             IOCLR1 = 0x00FF0000;
23             controlLeds <<= 1;
24             IOSET1 = controlLeds;
25          }
26
27       }
28   }
29 }
```

Figura 48 - Código fonte do primeiro ensaio com a placa MCB2130
Fonte: do autor

As linhas quatro e cinco são utilizadas para a declaração das variáveis utilizadas no programa. A linha seis é utilizada para habilitar a saída dos LED`s. A linha sete contém um laço infinito utilizando o comando *while* com o parâmetro “1”. Note que isso é extremamente importante para manter o programa ativo, caso contrário o programa teria o seu início e na seqüencialidade dos comandos, o seu fim. Nesse caso, é possível fazer uma analogia com as *threads*.

Na linha oito é feito um teste para verificar se o botão da placa de desenvolvimento está pressionado, para entender essa linha é necessário ter um conhecimento sobre binários e as suas operações lógicas. Exemplo, o valor de 0x00004000 em HEXA corresponde 1000 0000 0000 0000 em binário. É utilizado esse valor para testar se o bit 15 da constante IOPIN0 está habilitado ou não com a operação AND (&). Se o resultado dessa comparação for zero, então podemos dizer que o botão está pressionado. Segue um exemplo da comparação com resultado positivo na Tabela 23.

Tabela 23 - Identificação do pressionamento do botão INT1

I/O	HEXA	Binário
IOPIN0	0xE0028000	000000000000000
	0x00004000	100000000000000
	Resultado AND (&)	000000000000000

Fonte: do autor

A linha 11 é importante nessa aplicação. Ela é responsável por deixar o fluxo do sistema seguir somente quando o usuário libera o botão. Se não fosse implementada essa linha, não seria possível visualizar o ligar e desligar dos LED`s, pois isso aconteceria em frações de milissegundos. Na linha 12 existe um auto-incremento utilizado para verificar se o botão ficou pressionado em torno de um segundo, caso isso tenha ocorrido. Na linha 13 é testado esse campo de auto incremento, e, se o resultado for verdadeiro, é dado seqüência no programa. Essa rotina faz o efeito de se poder segurar o botão e os LED`s irem se movimentando na seqüência crescente e ao chegar ao oitavo LED voltar para o primeiro.

O controle de qual LED deve ligar e qual deve desligar encontra-se entre as linhas 17 e 25. Nessas linhas é verificado o LED atual ligado e na seqüência são desligados todos os LED`s e ligado o próximo da seqüência utilizando-se o deslocar de um bit para a esquerda. Abaixo seguem alguns valores utilizados para manipular os LED`s.

Tabela 24 - Configuração do acionamento dos LED`s

HEXA	Binário	Descrição
0x00FF0000	11111111000000000000000	Desliga os oito LED`s
0x0010000	00000001000000000000000	Liga o primeiro LED
0x00800000	10000000000000000000000	Liga o oitavo LED

Fonte: do autor

Seguindo a lógica apresentada acima, é possível fazer as mais diversas combinações para ligar e desligar os LED`s no kit de desenvolvimento da Keil Software.

Utilização de uma interrupção FIQ

O ensaio realizado tem como finalidade fazer o uso de interrupções e funcionando da seguinte maneira, toda vez que o usuário pressionar o botão INIT1 o sistema deve gerar uma interrupção fazendo com que o próximo LED da seqüência deva ser ligado e o anterior desligado, sendo que, quando for o último LED a ser desligado o primeiro LED deve ser aceso. O funcionamento desse ensaio é semelhante ao ensaio do acionamento do LED através do botão, porém ele utiliza interrupções. Nesse ensaio já foi utilizado o conceito de máquina de estados, a idéia é com que o programa principal não fique travado em um determinado ponto. A Figura 49 exibe o código fonte da aplicação.

```
01 #include <LPC213x.H>
02 unsigned int controlLeds  = 0x00800000;
03 unsigned int autoIncLeds  = 0x001;
04 unsigned int INTERRUPCAO_ACIONADA_FIQ = 0x00;
05 int main (void) {
06   IODIR1         = 0x00FF0000;
07
08   PINSEL0        = 0x20000000;
09   VICIntSelect   = 0x00008000;
10   VICIntEnable   = 0x00008000;
11   IOCLR1         = 0x00FF0000;
12   while (1) {
13        if (INTERRUPCAO_ACIONADA_FIQ == 0x01){
14              INTERRUPCAO_ACIONADA_FIQ = 0x00;
15              if(controlLeds == 0x00800000){
16                 controlLeds = 0x00010000;
17                 IOCLR1      = 0x00FF0000;
18                 IOSET1      = controlLeds;
19              }else{
20                 IOCLR1 = 0x00FF0000;
21                 controlLeds <<= 1;
22                 IOSET1 = controlLeds;
23              }
24        }
25   }
26 }
27 /* Função FIQ */
28 void FIQ_Handler (void) __fiq  {
29     INTERRUPCAO_ACIONADA_FIQ = 0x01;
30     EXTINT                   = 0x00000002;
31 }
```

Figura 49 - Código fonte utilizando interrupção FIQ
Fonte: do autor

As linhas dois e três têm a declaração das variáveis utilizadas para controle da movimentação dos LED`s. A linha quatro faz a declaração de uma variável de primeiro nível para controle da máquina de estados. Essa variável será utilizada no ciclo principal do programa. Na linha seis é ativado a variável IODIR1 que determina os LED`s como saída e na linha oito é ativado o botão um no bloco de seleção.

As linhas nove, dez e onze são utilizadas para configurar a interrupção. A linha nove é responsável por habilitar a interrupção FIQ, para isso basta configurar a variável VICIntSelect para 0x00008000 em hexadecimal, se converter esse valor para binário tem-se 1000000000000000, o bit 16 ativado, isso em outras palavras representa o canal da interrupção apresentado anteriormente na Tabela 1, lembrando que a contagem dos bits começa em zero. Então temos o bit 15 ativo e que nos representa o canal do INIT1. Para habilitar o canal VIC com a interrupção FIQ é necessário ativá-lo e para isso utiliza-se a variável *VICIntEnable* que deve ser configurada com o seguinte valor: 0x00008000. Nesse momento já está habilitada a interrupção.

Após a declaração das variáveis que irão habilitar a interrupção é necessário criar a função que irá responder por essa interrupção. Nas linhas 28, 29, 30 e 31 é definida a função que é executada quando a interrupção é ocorre. O código fonte dessa função é muito simples, é ativada a variável de controle da máquina de estados e na seqüência é limpo o registro que marca a interrupção. É importante lembrar que a execução de uma interrupção ocorre na borda de subida ou descida, isso quer dizer que, a interrupção desse exemplo somente será executada quando o usuário soltar o botão INIT1.

A diferença entre esse ensaio e o anterior está relacionada a questão de funcionalidade. No ensaio anterior o sistema ficava travado até que o usuário soltasse o botão INIT1 pressionado, porém nesse ensaio o sistema está sempre executando e quando ocorrer a interrupção o código é desviado para a função FIQ e após a conclusão da função o foco do sistema volta para a rotina principal.